(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
4 January 2001 (04.01.2001)

PCT

(10) International Publication Number
WO 01/01702 A1

(51) International Patent Classification⁷: H04N 7/68, G11B 20/18

(21) International Application Number: PCT/US00/14245

(22) International Filing Date: 24 May 2000 (24.05.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/342,322 29 June 1999 (29.06.1999) US

(71) Applicant: SONY ELECTRONICS, INC. [US/US]; 1 Sony Drive, Park Ridge, NJ 07656 (US).

(72) Inventors: KONDO, Tetsujiro; 12-1255 1-14 Tsumadakita, Atsugi-shi, Kanagawa-Prefecture 243 (JP). FUJIMORI, Yasuhiro; 11693 Westshore Court, Cupertino, CA 95014 (US). CAREY, William, Knox; 1355 McKendrie Street, San Jose, CA 95126 (US). CARRIG, James, J.; 1555 Minnesota Avenue, San Jose, CA 95125 (US).

(74) Agents: SOBRINO, Maria, E. et al.; Blakely, Sokoloff, Taylor & Zafman, 7th Floor, 12400 Wilshire Boulevard, Los Angeles, CA 90025-1026 (US).

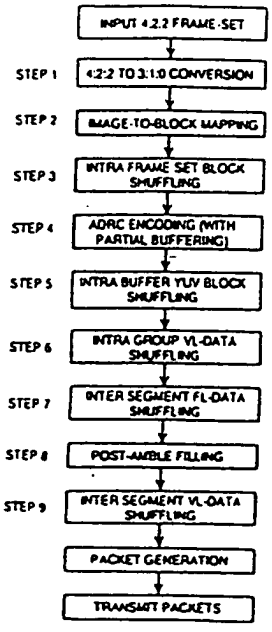
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *With international search report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: ERROR CONCEALMENT WITH PSEUDORANDOM INTERLEAVING DEPENDING ON A COMPRESSION PARAMETER



SIGNAL FLOWGRAPH IN THE ENCODER

(57) Abstract: Data is encoded to maximize subsequent recovery of lost or damaged compression parameters of encoded data. In one embodiment, at least one compression parameter is used to define a pseudorandom sequence and the data is shuffled using the pseudorandom sequence. In one embodiment, a bit reallocation process and code reallocation process are performed on the data to randomize the data.

WO 01/01702 A1

ERROR CONCEALMENT WITH PSEUDORANDOM INTERLEAVING DEPENDING ON A COMPRESSION PARAMETER

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to an encoding process that provides a robust error recovery due to data losses incurred during transmission of signals. More particularly, the present invention relates to a data shuffling method used in facilitating a robust error recovery.

2. Art Background

A number of techniques exist for reconstructing lost data due to random errors that occur during signal transmission or storage. However, these techniques cannot handle the loss of consecutive packets of data. Consecutive loss of packets of data is described in the art as burst error. Burst errors result in a reconstructed signal with such a degraded quality that it is easily apparent to the end user. Additionally, compression methodologies used to facilitate high speed communications compound the signal degradation caused by burst errors, thus adding to the degradation of the reconstructed signal. Examples of burst error loss affecting transmitted and/or stored signals may be seen in high definition television ("HDTV") signals, mobile telecommunication applications, as well as video storage technologies including video disk, compact disc and video cassette recorders (VCRs).

For example, the advent of HDTV has led to television systems with a much higher resolution than the current standards proposed by the National Television Systems Committee ("NTSC"). Proposed HDTV signals are predominantly digital. Accordingly, when a color television signal is converted for digital use it is common that the luminance and chrominance signals may be digitized using eight bits. Digital transmission of NTSC color television signals may require a nominal bit rate of about two-hundred and sixteen megabits per second. The transmission rate is greater for HDTV, which may nominally require about 1200 megabits per second. Such high transmission rates may be

well beyond the bandwidths supported by current wireless standards. Accordingly, an efficient compression methodology is required.

Compression methodologies also play an important role in mobile telecommunication applications. Typically, packets of data are communicated between remote terminals in mobile telecommunication applications. The limited number of transmission channels in mobile communications requires an effective compression methodology prior to the transmission of packets. A number of compression techniques are available to facilitate high transmission rates.

Adaptive Dynamic Range Coding ("ADRC") and Discrete Cosine Transform ("DCT") coding provide image compression techniques known in the art. Both techniques take advantage of the local correlation within an image to achieve a high compression ratio. However, an efficient compression algorithm may result in compounded error propagation because errors in an encoded signal are more prominent when subsequently decoded. This error multiplication may result in a degraded video image that is readily apparent to the user.

SUMMARY OF THE INVENTION

Data is encoded to enhance subsequent recovery of lost or damaged compression parameters of encoded data. In one embodiment, at least one compression parameter is used to define a pseudorandom sequence and the pseudurandom sequence is used to shuffle the data

BRIEF DESCRIPTION OF THE DRAWINGS

The objects, features and advantages of the present invention will be apparent to one skilled in the art in light of the following detailed description in which:

Figure 1a illustrates an embodiment of the processes of signal encoding, transmission, and decoding.

Figures 1b and 1c illustrate embodiments of signal encoding, transmission, and decoding implemented as software executed by a processor.

Figures 1d and 1e illustrate embodiments of circuits for shuffling and recovery of data.

Figure 2 illustrates one embodiment of a packet structure.

Figure 3 is a flow diagram illustrating one embodiment of the encoding process in accordance with the teachings of the present invention.

Figure 4 is a flow diagram illustrating one embodiment of the decoding process in accordance with the teachings of the present invention.

Figure 5 illustrates one embodiment of image-to-block mapping in accordance with the teachings of the present invention.

Figure 5a illustrates one embodiment of a shuffling pattern used in image-to-block mapping.

Figure 6 is an illustration of exemplary complementary and interlocking block structures.

Figures 7a, 7b, 7c, 7d illustrate one embodiment of shuffling patterns for Y blocks within a frame set.

Figure 8 is an illustration of one embodiment of cumulative DR distribution for Buffer 0.

Figure 8a is an illustration of one embodiment of a partial buffering process in accordance with the teachings of the present invention.

Figure 9 illustrates one embodiment of the intra buffer YUV block shuffling process in accordance with the teachings of the present invention.

Figure 10 illustrates one embodiment of the intra group VL-data shuffling process in accordance with the teachings of the present invention.

Figure 11 illustrates one embodiment of Qcode concatenation within a 3-block group in accordance with the teachings of the present invention.

Figure 11a illustrates one embodiment of Qcode concatenation for frame pairs including motion blocks in accordance with the teachings of the present invention.

Figure 12 illustrates one embodiment of pixel data error caused by a 1/6 burst error loss.

Figure 12a illustrates one embodiment of shuffling Qcodes and distributing Qcode bits in accordance with the teachings of the present invention.

Figure 12b illustrates one embodiment of pixel data error caused by a 1/6 burst error loss of redistributed Qcodes.

Figure 12c illustrates one embodiment of pixel data error caused by a 1/6 burst error loss of reassigned Qcodes.

Figure 12d illustrates one embodiment of a randomization process.

Figures 12e, 12f, 12g and 12h are examples of randomization processes.

Figure 13 illustrates one embodiment of MIN shuffling in accordance with the teachings of the present invention.

Figure 13a illustrates one embodiment of Motion Flag shuffling and of a fixed length data loss in one frame pair.

Figure 14 illustrates one embodiment of a modular shuffling.

Figure 14a illustrates one embodiment of a modular shuffling result and the fixed length data loss associated with the modular shuffling.

Figure 14b illustrates an alternate embodiment of a modular shuffling result and the fixed length data loss associated with the modular shuffling.

Figure 14c illustrates an alternate embodiment of a modular shuffling result and the fixed length data loss associated with the modular shuffling.

Figure 15 illustrates one embodiment of variable length data buffering in a frame set.

Figure 16 illustrates one embodiment of inter segment VL-data shuffling in accordance with the teachings of the present invention.

DETAILED DESCRIPTION

The present invention provides a system and method for the shuffling of a signal stream to provide for a robust error recovery. In the following description, for purposes of explanation, numerous details are set forth in order to provide a thorough understanding of the present invention. However, it will be apparent to one skilled in the art that these specific details are not required in order to practice the present invention. In other instances, well known electrical

structures and circuits are shown in block diagram form in order not to obscure the present invention unnecessarily.

The signal processing methods and structures are described in the context of one embodiment in which the signals are Adaptive Dynamic Range Coding (ADRC) encoded images, and more particularly to the recovery of lost or damaged (lost/damaged) compression parameters such as dynamic range (DR) and minimum value (MIN). However, it is contemplated that the present invention is not limited to ADRC encoding and the particular compression parameters generated; rather it will be apparent that the present invention is applicable to different compression technologies, different types of correlated data, including, but not limited to, sound data and the like, and different compression parameters, including, but not limited to, the number of bits used to encode data (Qbit), maximum value (MAX) and central value (CEN), which may be used in ADRC processes.

In addition, the present invention is applicable to different types of ADRC processes including edge-matching and non edge-matching ADRC. For further information regarding ADRC, see "Adaptive Dynamic Range Coding Scheme for Future HDTV Digital VTR", Kondo, Fujimori, Nakaya, Fourth International Workshop on HDTV and Beyond, September 4-6, 1991, Turin, Italy. ADRC has been established as a feasible real-time technique for coding and compressing images in preparation for constant bit-rate transmission.

In the above paper, three different kinds of ADRC are explained. These are achieved according to the following equations:

Non-edge-matching ADRC:

$$DR = MAX - MIN + 1$$

$$q = \left\lfloor \frac{(x - MIN + 0.5) \cdot 2^Q}{DR} \right\rfloor$$

$$x' = \left\lfloor \frac{(q + 0.5) \cdot DR}{2^Q} + MIN \right\rfloor$$

Edge-matching ADRC:

$$DR = MAX - MIN$$

$$q = \left\lfloor \frac{(x - MIN) \cdot (2^Q - 1)}{DR} + 0.5 \right\rfloor$$

$$x' = \left\lfloor \frac{q \cdot DR}{2^Q - 1} + MIN + 0.5 \right\rfloor$$

Multi-stage ADRC:

$$DR = MAX - MIN + 1$$

$$q = \left\lfloor \frac{(x - MIN + 0.5) \cdot 2^Q}{DR} \right\rfloor$$

$$x' = \left\lfloor \frac{(q + 0.5) \cdot DR}{2^Q} + MIN \right\rfloor$$

Where MAX' is the averaged value of x' in the case of $q = 2^Q - 1$;

MIN' is the averaged value of x' in the case of $q = 0$; and

$$DR' = MAX' - MIN'$$

$$q = \left\lfloor \frac{(x - MIN') \cdot (2^Q - 1)}{DR'} + 0.5 \right\rfloor$$

$$x' = \left\lfloor \frac{q \cdot DR'}{(2^Q - 1)} + MIN' + 0.5 \right\rfloor$$

where MAX represents the maximum level of a block, MIN represents the minimum level of a block, x represents the signal level of each sample, Q represents the number of quantization bits, q represents the quantization code (encoded data), x' represents the decoded level of each sample, and the square brackets $\lfloor \cdot \rfloor$ represent a truncation operation performed on the value within the square brackets.

The signal encoding, transmission, and subsequent decoding processes are generally illustrated in **Figure 1a**. Signal 100 is a data stream input to Encoder 110. Encoder 110 follows the Adaptive Dynamic Range Coding ("ADRC") compression algorithm and generates Packets 1, . . . N for transmission along Transmission Media 135. Decoder 120 receives Packets 1, . . .

N from Transmission Media 135 and generates Signal 130. Signal 130 is a reconstruction of Signal 100.

Encoder 110 and Decoder 120 can be implemented a variety of ways to perform the functionality described herein. In one embodiment, Encoder 110 and/or Decoder 120 may be embodied as software stored on media and executed by a general purpose or specifically configured computer system, typically including a central processing unit, memory and one or more input/output devices and co-processors, as shown in **Figures 1b and 1c**. Alternately, the Encoder 110 and/or Decoder 120 may be implemented as logic to perform the functionality described herein, as shown in **Figures 1d and 1e**. In addition, Encoder 110 and/or Decoder 120 can be implemented as a combination of hardware, software or firmware.

Embodiments of the circuits are shown in **Figures 1b and 1c**. The methods described herein may be implemented on a specially configured or general purpose processor system 170. Instructions are stored in memory 190 and accessed by processor 175 to perform many of the steps described herein. Input 180 receives the input bitstream and forwards the data to processor 175. Output 185 outputs the data. In **Figure 1b**, the output may consist of the encoded data. In **Figure 1c**, the output may consist of the decoded data, such as image data decoded according to the methods described, sufficient to drive an external device such as display 195.

In one embodiment, Signal 100 may be a color video image comprising a sequence of video frames, each frame including information representative of an image in an interlaced video system. Each frame is composed of two fields, wherein one field contains data of the even lines of the image and the other field containing the odd lines of the image. The data includes pixel values that describe the color components of a corresponding location in the image. For example, in the present embodiment, the color components consist of the luminance signal Y, and color difference signals U, and V. It is readily apparent the process of the present invention can be applied to signals other than interlaced video signals. Furthermore, it is apparent that the present invention is

not limited to implementations in the Y, U, V color space, but can be applied to images represented in other color spaces.

In alternate embodiments, Signal 100 may be, for example, two-dimensional static images, hologram images, three-dimensional static images, video, two-dimensional moving images, three dimensional moving images, monaural sound, or N-channel sound.

Referring back to Figure 1a, Encoder 110 divides the Y, U, and V signals and processes each group of signals independently in accordance with the ADRC algorithm. The following description, for purposes of simplifying the discussion, describes the processing of the Y signal; however, the encoding steps may be replicated for the U and V signals.

In one embodiment, Encoder 110 groups Y signals across two subsequent frames, referred to herein as a frame pair, of Signal 100 into three dimensional blocks ("3D") blocks. In an alternate embodiment, a 3D block is generated from grouping two 2D blocks from the same localized area across a given frame pair, wherein a two dimensional 2D block is created by grouping localized pixels within a frame or a field. It is contemplated that the process described herein can be applied to different block structures. The grouping of signals will be further described in the image-to-block mapping section below.

In one embodiment, for a given 3D block, Encoder 110 calculates whether there is a change in pixel values between the 2D blocks forming the 3D block. A Motion Flag is set if there are substantial changes in values. As is known in the art, use of a Motion Flag allows Encoder 110 to reduce the number of quantization codes when there is localized image repetition within each frame pair. Encoder 110 also detects the maximum pixel intensity value ("MAX") and the minimum pixel intensity value ("MIN") within a 3D block. Using values MAX and MIN, Encoder 110 calculates the dynamic range ("DR") for a given 3D block of data. For one embodiment, DR = MAX - MIN + 1 in the case of non-edge-matching ADRC. For edge-matching ADRC, DR = MAX - MIN. In some embodiments the encoder may also determine a central value (CEN) that has a

value between MAX and MIN. In one embodiment, CEN may be determined as CEN = MIN + DR/2.

In an alternative embodiment, Encoder 110 encodes signals on a frame by frame basis for a stream of frames representing a sequence of video frames. In another embodiment, Encoder 110 encodes signals on a field by field basis for a stream of fields representing a sequence of video fields. Accordingly, Motion Flags are not used and 2D blocks may be used to calculate the MIN, MAX, CEN and DR values.

In one embodiment, Encoder 110 references the calculated DR against a threshold table of DR threshold values and corresponding Qbit values to determine the number of quantization bits ("Qbits") used to encode pixels within the block corresponding to the DR. Encoding of a pixel results in a quantization code ("Qcode"). The Qcodes are the relevant compressed image data used for storage or transmission purposes.

In one embodiment, the Qbit selection is derived from the DR of a 3D block. Accordingly, all pixels within a given 3D block are encoded using the same Qbit, resulting in a 3D encoded block. The collection of Qcodes, MIN, Motion Flag, and DR for a 3D encoded block is referred to as a 3D ADRC block. Alternately, 2D blocks are encoded and the collection of Qcodes, MIN, and DR for a given 2D block results in 2D ADRC blocks. As noted earlier, the MAX value and CEN value may be used in place of the MIN value.

A number of threshold tables can be implemented. In one embodiment, the threshold table consists of a row of DR threshold values. A Qbit corresponds to the number of quantization bits used to encode a range of DR values between two adjacent DRs within a row of the threshold table. In an alternative embodiment, the threshold table includes multiple rows and selection of a row depends on the desired transmission rate. Each row in the threshold table is identified by a threshold index. A detailed description of one embodiment of threshold selection is described below in the discussion of partial buffering. A further description of an example of ADRC encoding and buffering is disclosed in US Patent No. 4,722,003 entitled "High Efficiency Coding Apparatus" and US

Patent No. 4,845,560 also entitled "High Efficiency Coding Apparatus", assigned to the assignee of the present invention.

Here forth the Qcodes are sometimes referred to as variable length data ("VL-data"). In addition, the DR, MIN, MAX, CEN and Motion Flag are referred to as block attributes. Selected block attributes, together with the threshold index, constitute the fixed length data ("FL-data"), also referred to herein as compression parameters. Furthermore, in view of the above discussion, the term block attribute describes a parameter associated with a component of a signal element, wherein a signal element includes multiple components.

In an alternate embodiment, the FL-data includes a Qbit code. The advantage is that the Qbit information does not have to be derived from the DR during the decoding process. Thus, if the DR information is lost or damaged, the Qbit information can still be determined from the Qbit code. Furthermore, if the Qbit code is lost or damaged, the Qbit information can be derived from DR. Thus the requirement to recover the DR and Qbit is reduced.

The disadvantage to including the Qbit code is the additional bits to be transmitted for each ADRC block. However, in one embodiment, Qbit codes for groups of ADRC blocks are combined, for example, in accordance with a function such as addition or concatenation. For example, if ADRC blocks are grouped in threes and if the Qbit values for each ADRC block are respectively 3, 4 and 4, the summed value that is encoded into the FL-data is 11. Thus the number of bits required to represent the sum is less than the number of bits required to represent each individual value and undamaged Qbit values of the group can be used to determine the Qbit value without performing a Qbit recovery process such as the one described subsequently.

Other embodiments are also contemplated. For example, Motion Flag data may also be encoded. A tag with Qbit and Motion Flag data can be generated and used to reference a table of codes. The configuration and function of the coding can vary according to application.

An advantage of not including the Qbit code value in the FL-data is that no additional bits are need be transmitted for each ADRC block. A disadvantage

of not including the Qbit value is that, if the DR is lost or damaged during transmission or storage, the Qcodes cannot be easily recovered. The ADRC decoder must determine how many bits were used to quantize the block without relying on any DR information.

However, as will be described below, recovery of a lost or damaged Qbit value may be enhanced by randomization or shuffling of the VL-data. One embodiment of a circuit for shuffling to provide for a robust error recovery is shown in **Figure 1d**. An input signal is received and VL-data shuffling logic 144 generates randomized Qcodes based upon the encoded and/or shuffled data. It should be noted that the output from the VL-data shuffling logic 144 may be precoded or further encoded as discussed herein.

Figure 1e illustrates an embodiment of a circuit for recovering lost or damaged values such as compression parameters. An input signal is received and VL-data deshuffling logic 150 derandomizes the Qcodes from the input bitstream and recovers lost or damaged constants. The output signal from the VL-data deshuffling logic 150 may be further decoded and/or deshuffled as described herein.

In some embodiments, as will be discussed below, a pseudorandom sequence may be generated by, stored in or otherwise accessed by the shuffling logic 144 and deshuffling logic 150.

Frames, block attributes, and VL-data describe a variety of components within a video signal. The boundaries, location, and quantity of these components are dependent on the transmission and compression properties of a video signal. In the present embodiment, these components are varied and shuffled within a bitstream of the video signal to ensure a robust error recovery during transmission losses.

For illustrative purposes, the following description provides for a 1/6 consecutive packet transmission loss tolerance, pursuant to an ADRC encoding and shuffling of a video signal. Accordingly, the following definition and division of components exist for one embodiment. Other embodiments also are contemplated. A data set includes a partition of data of a video or other type of

data signal. Thus, in one embodiment, a frame set is a type of data set that includes one or more consecutive frames. A segment includes a memory with the capacity to store a one-sixth division of the Qcodes and block attributes included in a frame set. Further, a buffer includes a memory with the capacity to store a one-sixtieth division of the Qcodes and block attributes included in a frame set. The shuffling of data is performed by interchanging components within segments and/or buffers. Subsequently, the data stored in a segment is used to generate packets of data for transmission. Thus, in the following description if a segment is lost all the packets generated from the segment are lost during transmission. Similarly, if a fraction of a segment is lost then a corresponding number of packets generated from the segment are lost during transmission.

Although, the following description refers to a 1/6 consecutive packet loss for data encoded using ADRC encoding, it is contemplated that the methods and apparatus described herein are applicable to a design of a 1/n consecutive packets loss tolerance coupled to a variety of encoding/decoding schemes.

Figure 2 illustrates one embodiment of Packet Structure 200 used for the transmission of the data across point-to-point connections as well as networks. Packet Structure 200 is generated by Encoder 110 and is transmitted across Transmission Media 135. For one embodiment, Packet Structure 200 comprises five bytes of header information, eight DR bits, eight MIN bits, a Motion Flag bit, a five bit threshold index, and 354 bits of Qcodes. In an alternate embodiment, the MIN bits may be replaced with CEN bits. The packet structure described herein is illustrative and may typically be implemented for transmission in an asynchronous transfer mode ("ATM") network. However, the present invention is not limited to the packet structure described and a variety of packet structures that are used in a variety of networks can be utilized.

As noted earlier, Transmission Media (e.g., media) 135 is not assumed to provide error-free transmission and therefore packets may be lost or damaged. As noted earlier, conventional methods exist for detecting such loss or damage, but substantial image degradation will generally occur. The system and

methods of the present invention therefore teach source coding to provide robust recovery from such loss or damage. It is assumed throughout the following discussion that a burst loss, that is the loss of several consecutive packets, is the most probable form of error, but some random packet losses might also occur.

To ensure a robust recovery for the loss of one or more consecutive packets of data, the system and methods of the present invention provide multiple level shuffling. In particular, the FL-data and the VL-data included in a transmitted packet comprise data from spatially and temporally disjointed locations of an image. Shuffling data ensures that any burst error is scattered and facilitates error recovery. As will be described below, the shuffling allows recovery of block attributes and Qbit values.

Data Encoding/Decoding

Figure 3 is a flow diagram illustrating one embodiment of the encoding process performed by Encoder 110. Figure 3 further describes an overview of the shuffling process used to ensure against image degradation and to facilitate a robust error recovery.

In step one of Figure 3, an input frame set, also referred to as a display component, is decimated to reduce the transmission requirements. The Y signal is decimated horizontally to three-quarters of its original width and the U and V signals are each decimated to one-half of their original height and one-half of their original width. This results in a 3:1:0 video format with 3960 Y blocks, 660 U blocks and 660 V blocks in each frame pair. As noted earlier, the discussion will describe the processing of Y signals; however, the process is applicable to the U and V signals. At step two, the two Y frame images are mapped to 3D blocks. At step three, 3D blocks are shuffled. At step four, ADRC buffering and encoding is used. At step five, encoded Y, U and V blocks are shuffled within a buffer.

At step six, the VL-data for a group of encoded 3D blocks and their corresponding block attributes are shuffled. At step seven, the FL-data is shuffled across different segments. At step eight, post-amble filling is performed

in which variable space at the end of a buffer is filled with a predetermined bitstream. At step nine, the VL-data is shuffled across different segments.

For illustrative purposes the following shuffling description provides a method for manipulation of pixel data before and after encoding. For an alternative embodiment, independent data values are shuffled/deshuffled via hardware. In particular, the hardware maps the address of block values to different addresses to implement the shuffling/deshuffling process. However, address mapping is not possible for data dependent values because shuffling has to follow the processing of data. The intra group VL-data shuffling described below includes the data dependent values. Further, for illustrative purposes the following shuffling description occurs on discrete sets of data. However, for alternative embodiments a signal is defined based on multiple data levels ranging from bits, to pixels, and to frames. Shuffling is possible for each level defined in the signal and across different data levels of the signal.

Figure 4 is a flow diagram illustrating one embodiment of decoding process performed by Decoder 120. In one embodiment, the conversion and de-shuffling processes may be the inverse of the processes represented in **Figure 3**.

Image-to-Block Mapping

In the present embodiment, a single frame typically comprises 5280 2D blocks wherein each 2D block comprises 64 pixels. Thus, a frame pair comprises 5280 3D blocks as a 2D block from a first frame and a 2D block from a subsequent frame are collected to form a 3D block.

Image-to-block mapping is performed for the purpose of dividing a frame or frame set of data into 2D blocks or 3D blocks respectively. Moreover, image-to-block mapping includes using a complementary and/or interlocking pattern to divide pixels in a frame to facilitate robust error recovery during transmission losses. However, to improve the probability that a given DR value is not too large, each 2D block is constructed from pixels in a localized area.

Figure 5 illustrates one embodiment of an image-to-block mapping process for an exemplary 16 pixel section of an image. Image 500 comprises 16 pixels forming a localized area of a single frame. Each pixel in Image 500 is

represented by an intensity value. For example, the pixel in the top left hand side of the image has an intensity value equal to 100 whereas the pixel in the bottom right hand side of the image has an intensity value of 10.

In one embodiment, pixels from different areas of Image 500 are used to create 2D Blocks 510, 520, 530, and 540. 2D Blocks 510, 520, 530, and 540 are encoded, shuffled (as illustrated below), and transmitted. Subsequent to transmission, 2D Blocks 510, 520, 530, and 540 are recombined and used to form Image 550. Image 550 is a reconstruction of Image 500.

To ensure accurate representation of Image 500 despite a possible transmission loss, **Figure 5** is an interlocking complementary block structure, one embodiment of which is illustrated in **Figure 5**, is used to reconstruct Image 550. In particular, the pixel selection used to create 2D Blocks 510, 520, 530, and 540 ensures that a complementary and/or interlocking pattern is used to recombine the blocks when Image 550 is reconstructed. Accordingly, when a particular 2D block's attribute is lost during transmission, contiguous sections of Image 550 are not distorted during reconstruction. For example, as illustrated in Figure 5 the DR of 2D Block 540 is lost during data transmission. However, during reconstruction of Image 550, the decoder utilizes multiple neighboring pixels of neighboring blocks through which a DR can be recovered for the missing DR of 2D Block 540. In addition, as will be subsequently described, the combination of complementary patterns and shifting increases the number of neighboring pixels, preferably maximizing the number of neighboring pixels that originate from other blocks, significantly improving DR and MIN recovery.

Figure 5a illustrates one embodiment of a shuffling pattern used to form 2D blocks in one embodiment of the image-to-block mapping process. An image is decomposed into two sub-images, Sub-Image 560 and Sub-Image 570, based on alternating pixels. Rectangular shapes are formed in Sub-Image 560 to delineate the 2D block boundaries. For purposes of discussion, the 2D blocks are numbered 0, 2, 4, 7, 9, 11, 12, 14, 16, 19, 21, and 23. Tile 565 illustrates the pixel distribution for a 2D block within Sub-Image 560.

In Sub-Image 570, the 2D block assignment is shifted by eight pixels horizontally and four pixels vertically. This results in a wrap around 2D block assignment and overlap when Sub-Images 560 and 570 are combined during reconstruction. The 2D blocks are numbered 1, 3, 5, 6, 8, 10, 13, 15, 17, 18, 20, and 22. Tile 575 illustrates the pixel distribution for a 2D block within Sub-Image 570. Tile 575 is the complementary structure of Tile 565. Accordingly, when a particular block's attribute is lost during transmission, neighboring pixels through which a block attribute can be recovered for the missing 2D block exists. Additionally, an overlapping 2D block of pixels with a similar set of block attributes exist. Therefore, during reconstruction of the image the decoder has multiple neighboring pixels from adjacent 2D blocks through which a lost block attribute can be recovered.

Figure 6 illustrates other complementary and interlocking 2D block structures. Other structures may also be utilized. Similar to Figure 5, these 2D block structures illustrated in Figure 6, ensure surrounding 2D blocks are present despite transmission losses for a given 2D block. However, Patterns 610a, 610b, and 610d use horizontal and/or vertical shifting during the mapping of pixels to subsequent 2D blocks. Horizontal shifting describes shifting the tile structure in the horizontal direction a predetermined number of pixels prior to beginning a new 2D block boundary. Vertical shifting describes shifting the tile structure in the vertical direction a predetermined number of pixels prior to beginning a new 2D block boundary. In application, horizontal shifting only may be applied, vertical shifting may only be applied, or a combination of horizontal and vertical shifting may be applied.

Pattern 610a illustrates a spiral pattern used for image-to-block mapping. The spiral pattern follows a horizontal shifting to create subsequent 2D blocks during the image-to-block mapping process. Patterns 610b and 610d illustrate complementary patterns wherein pixel selection is moved by a horizontal and vertical shifting to create subsequent 2D blocks during the image-to-block mapping process. Further, Patterns 610b and 610d illustrate alternating offsets on pixels selection between 2D blocks. Pattern 610c illustrates using an irregular

sampling of pixels to create a 2D block for image-to-block mapping. Accordingly, the image-to-block mapping follows any mapping structure provided a pixel is mapped to a 2D block only once.

Figure 5, Figure 5a and Figure 6 describe image-to-block mapping for 2D block generation. It is readily apparent that the processes are applicable to 3D blocks. As described above, 3D block generation follows the same boundary definition as a 2D block, however the boundary division extends across a subsequent frame resulting in a 3D block. In particular, a 3D block is created by collecting the pixels used to define a 2D block in a first frame together with pixels from a 2D block in a subsequent frame. In one embodiment, both pixels in the 2D block from the first frame and the 2D block from the subsequent frame are from the exact same location.

Intra Frame Set Block Shuffling

The pixels values for a given image are closely related for a localized area. However, in another area of the same images the pixel values may have significantly different values. Thus, subsequent to encoding the DR and MIN values for spatially close 2D or 3D blocks in a section of an image have similar values, whereas the DR and MIN values for blocks in another section of the image may be significantly different. Accordingly, when buffers are sequentially filled with encoded data from spatially close 2D or 3D blocks of an image, a disproportionate usage of buffer space occurs. Intra frame set block shuffling occurs prior to ADRC encoding and includes shuffling the 2D or 3D blocks generated during the image-to-block mapping process. This shuffling process ensures an equalized buffer usage during a subsequent ADRC encoding.

Figures 7a - 7d illustrate one embodiment of shuffling 3D Y-blocks. The 3D Y-blocks in Figures 7a-7d are generated from applying the image-to-block mapping process described above to a frame pair containing only Y signals. The 3D Y-blocks are shuffled to ensure that the buffers used to store the encoded frame pair contain 3D Y-blocks from different parts of the frame pair. This leads to similar DR distribution during ADRC encoding. A similar DR distribution within each buffer leads to consistent buffer utilization.

Figure 7a -7d also illustrate 3D block shuffling using physically disjointed 3D blocks to ensure that transmission loss of consecutive packets results in damaged block attributes scattered across the image, as opposed to a localized area of the image.

The block shuffling is designed to widely distribute block attributes in the event of small, medium, or large, burst packet losses occur. In the present embodiment, a small burst loss is thought of as one where a few packets are lost; a medium loss is one in which the amount of data that can be held in one buffer is lost; and a large loss is one in which the amount of data that can be held in one segment is lost. During the 3D block shuffling each group of three adjacent blocks are selected from relatively remote parts of the image. Accordingly, during the subsequent intra group VL-data shuffling (to be detailed later), each group is formed from 3D blocks that have differing statistical characteristics. Distributed block attribute losses allow for a robust error recovery because a damaged 3D block is surrounded by undamaged 3D blocks and the undamaged 3D blocks can be used to recover lost data.

Figure 7a illustrates a frame pair containing 66 3D Y-blocks in the horizontal direction and 60 3D Y-blocks in the vertical direction. The 3D Y-blocks are allocated into Segments 0 - 5. As illustrated, the 3D Y-block assignment follows a two by three column section such that one 3D Y-block from each section is associated with a segment. Thus, if no further shuffling is performed and a burst loss of the first 880 packets occurs, all the block attributes associated with Segment 0 are lost. However, as later described, FL-data shuffling is performed to further disperse block attribute losses.

Figure 7b illustrates the scanning order of 3D Y-blocks numbered "0" used to enter into Segment 0. Each "0" 3D Y-block of Figure 7a is numbered 0, 1, 2, 3, ..., 659 to illustrate their location in the stream that is inputted into Segment 0. Using the block numbering to allocate segment assignment the remaining 3D Y-blocks are inputted into Segments 1 - 5, thus resulting in a frame pair shuffled across multiple segments.

Figure 7c illustrates the 660 3D Y-blocks comprising one segment. The 3D Y-blocks numbered 0 - 65 are inputted into Buffer 0. Similarly the 3D Y-blocks adjacent to the numbered 3D Y-blocks are inputted into Buffer 1. The process is repeated to fill Buffers 2 - 9. Accordingly, damage to a buffer during data transmission results in missing 3D Y-blocks from different parts of the image.

Figure 7d illustrates the final ordering of the "0" 3D Y-blocks across a buffer. 3D Y-blocks 0, 1, and 2 occupy the first three positions in the buffer. The process is repeated for the rest of the buffer. Accordingly, loss of three 3D Y-blocks during data transmission results in missing 3D Y-blocks from distant locations within the image.

Figures 7a-d illustrate one embodiment of 3D block distributions for 3D Y-blocks of a frame set. In alternative embodiments, however, 3D block distributions for 3D U-blocks and 3D V-blocks are available. The 3D U-blocks are generated from applying the image-to-block mapping process, described above, to a frame set containing only U signals. Similarly, 3D V-blocks are generated from applying the image-to-block mapping process to a frame set containing only V signals. Both the 3D U-block and the 3D V-block follow the 3D Y-block distribution described above. However, as previously described, the number of 3D U-blocks and 3D V-blocks each have a 1:6 proportion to 3D Y-blocks.

Figures 7a-d are used to illustrate one embodiment of intra frame set block shuffling for a Y signal such that burst error of up to 1/6 of the packets lost during transmission is tolerated and further ensures equalized buffer use. It will be appreciated by one skilled in the art that segment, buffer, and ADRC block assignments can be varied to ensure against 1/n burst error loss or to modify buffer utilization.

Partial Buffering

As illustrated in **Figure 3**, the ADRC encoding and buffering processes occur in step four. Dependent on the encoding technique, 2D or 3D blocks generated during the image-to-block mapping process are encoded resulting in 2D or 3D ADRC blocks. A 3D ADRC block, contains Qcodes, a MIN value, a

Motion Flag, and a DR. Similarly, a 2D ADRC block contains Qcodes, a MIN, and a DR. A 2D ADRC block, however, does not include a Motion Flag because the encoding is performed on a single frame or a single field.

A number of buffering techniques are found in the prior art (see for example, High Efficiency Coding Apparatus, U.S. Patent 4,845,560 of Kondo et. al. and High Efficiency Coding Apparatus, U.S. Patent 4,722,003 of Kondo). Both High Efficiency Coding Apparatus patents are hereby incorporated by reference.

The partial buffering process set forth below, describes an innovative method for determining the encoding bits used in ADRC encoding. In particular, partial buffering describes a method of selecting threshold values from a threshold table designed to provide a constant transmission rate between remote terminals while restricting error propagation. In an alternative embodiment, the threshold table is further designed to provide maximum buffer utilization. In one embodiment, a buffer is a memory that stores a one-sixtieth division of encoded data from a given frame set. The threshold values are used to determine the number of Qbits used to encode the pixels in 2D or 3D blocks generated from the image-to-block mapping process previously described.

The threshold table includes rows of threshold values, also referred to as a threshold set, and each row in the threshold table is indexed by a threshold index. In one embodiment, the threshold table is organized with threshold sets that generate a higher number of Qcode bits located in the upper rows of the threshold table. Accordingly, for a given buffer having a predetermined number of bits available, Encoder 110 moves down the threshold table until a threshold set that generates less than a predetermined number of bits is encountered. The appropriate threshold values are used to encode the pixel data in the buffer.

In one embodiment, a transmission rate of no more than 30 Mbps is desired. The desired transmission rate results in 31,152 bits available for VL-data storage in any given buffer. Accordingly, for each buffer a cumulative DR distribution is computed and a threshold set is selected from the threshold table to encode the pixels in 3D or 2D blocks into VL-data.

Figure 8 illustrates one embodiment of selected threshold values and the DR distribution for Buffer 0. The vertical axis of Figure 8 includes the cumulative DR distribution. For example, the value "b" is equal to the number of 3D or 2D blocks whose DR is greater than or equal to L_3. The horizontal axis includes the possible DR values. In one embodiment, DR values range from 0 to 255. Threshold values L_4, L_3, L_2, and L_1 describe a threshold set used to determine the encoding of a buffer.

In one embodiment, all blocks stored in Buffer 0 are encoded using threshold values L_4, L_3, L_2, and L_1. Accordingly, blocks with DR values greater than L_4 have their pixel values encoded using four bits. Similarly, all pixels belonging to blocks with DR values between L_3 and L_4 are encoded using three bits. All pixels belonging to blocks with DR values between L_2 and L_3 are encoded using two bits. All pixels belonging to blocks with DR values between L_1 and L_2 are encoded using one bit. Finally, all pixels belonging to blocks with DR values smaller than L_1 are encoded using zero bits. L_4, L_3, L_2, and L_1 are selected such that the total number of bits used to encode all the blocks in Buffer 0 is as close as possible to a limit of 31,152 bits without exceeding the limit of 31,152.

Figure 8a illustrates the use of partial buffering in one embodiment. Frame 800 is encoded and stored in Buffers 0 - 59. Provided a transmission error inhibits data recovery, the decoding process is stalled for Frame 800 until error recovery is performed on the lost data. However, partial buffering restricts the error propagation within a buffer, thus allowing decoding of the remaining buffers. In one embodiment, a transmission error inhibits the Qbit and Motion Flag recovery for Block 80 in Buffer 0. Partial buffering limits the error propagation to the remaining blocks within Buffer 0. Error propagation is limited to Buffer 0 because the end of Buffer 0 and the beginning of Buffer 1 are known due to the fixed buffer length. Accordingly, Decoder 120 can begin processing of blocks within Buffer 1 without delay. Additionally, the use of different threshold sets to encode different buffers allows Encoder 110 to

maximize/control the number of Qcodes bits included in a given buffer, thus allowing a higher compression ratio. Furthermore, the partial buffering process allows for a constant transmission rate because Buffers 0 - 59 consist of a fixed length.

In one embodiment, a buffer's variable space is not completely filled with Qcode bits because a limited number of threshold sets exist. Accordingly, the remaining bits in the fixed length buffer are filled with a predetermined bitstream pattern referred to as a post-amble. As will be described subsequently, the post-amble enables bidirectional data recovery because the post-amble delineates the end of the VL-data prior to the end of the buffer.

Intra Buffer YUV Block Shuffling

Y, U, and V, signals each have unique statistical properties. To improve the Qbit and Motion Flag recovery process (described below) the Y, U, and V signals are multiplexed within a buffer. Accordingly, transmission loss does not have a substantial effect on a specific signal.

Figure 9 illustrates one embodiment of the intra buffer YUV block shuffling process in which YUV ADRC blocks are derived from the Y, U, and V signals respectively. Buffer 900 illustrates the ADRC block assignments after intra frame set block shuffling. Buffer 900 comprises 66 Y-ADRC blocks followed by 11 U-ADRC blocks which are in turn followed by 11 V-ADRC blocks. Buffer 910 shows the YUV ADRC block organization after intra buffer YUV block shuffling. As illustrated, three Y-ADRC blocks are followed by a U-ADRC block or three Y-ADRC blocks are followed by a V-ADRC block. Intra buffer YUV block shuffling reduces similarity between adjacent block's bitstreams within the buffer. Alternative embodiments of intra buffer YUV block shuffling with a different signal, i.e., YUV ratios or other color spaces are possible dependent on the initial image format.

Intra Group VL-Data Shuffling

In one embodiment, Intra group VL-data shuffling comprises three processing steps. The three processing steps include Qcode concatenation, Qcode reassignment, and randomizing concatenated Qcodes. **Figure 10**

illustrates one embodiment of intra group VL-data shuffling wherein three processing steps are applied consecutively to Qcodes stored in a buffer. In alternative embodiments, one or more processing steps discussed herein may be applied to perform intra group VL-data shuffling. Each processing step independently assists in the error recovery of data lost during transmission. Accordingly, each processing step is described independently.

1. Qcode concatenation

Qcode concatenation ensures that groups of ADRC blocks are decoded together. Group decoding facilitates error recovery because additional information is available from neighboring blocks during the data recovery process detailed below. For one embodiment, Qcode concatenation is applied independently to each group of three ADRC blocks stored in a buffer. In an alternative embodiment, a group includes ADRC block(s) from different buffers. The concatenation of Qcodes across three ADRC blocks is described as generating one concatenated ADRC tile. **Figure 11** and **Figure 11a** illustrate one embodiment of generating concatenated ADRC tiles.

Figure 11 illustrates one embodiment of generating a concatenated ADRC tile from 2D ADRC blocks. Specifically, the concatenation is performed for each Qcode (q_0 - q_{63}) included in 2D ADRC Blocks 0, 1, and 2 resulting in the sixty four Qcodes of Concatenated ADRC Tile A. For example, the first Qcode $q_{0,0}$ (0th quantized value) of 2D ADRC Block 0 is concatenated to the first Qcode $q_{0,1}$ of 2D ADRC Block 1. The two concatenated Qcodes are in turn concatenated to the first Qcode $q_{0,2}$ of 2D ADRC Block 2, thus resulting in Q_0 of Concatenated ADRC Tile A. The process is repeated until Q_{63} is generated. Alternatively, the generation of Q_i in Concatenated ADRC Tile A is described by the equation

$$Q_i = [\, q_{i,0} , q_{i,1} , q_{i,2} \,] \qquad i = 0, 1, 2, \ldots 63$$

Additionally, associated with each Q_i in Concatenated ADRC Tile A there is a corresponding number of N bits that represents the total number of bits concatenated to generate a single Q_i.

Figure 11a illustrates one embodiment of generating a concatenated ADRC tile from frame pairs including motion blocks. A motion block is a 3D ADRC block with a set Motion Flag. The Motion Flag is set when a predetermined number of pixels within two 2D blocks structure created by image-to-block mapping process described earlier, change in value between a first frame and a subsequent frame. In an alternative embodiment, the Motion Flag is set when the maximum value of each pixel change between the 2D block of a first frame and a subsequent frame exceeds a predetermined value. In contrast, non-motion (i.e., stationary) block includes a 3D ADRC block with a Motion Flag that is not set. The Motion Flag remains un-set when a predetermined number of pixels within the two 2D blocks of a first frame and a subsequent frame do not change in value. In an alternative embodiment, the Motion Flag remains un-set when the maximum value of each pixel change between a first frame and a subsequent frame does not exceed a predetermined value.

A motion block includes Qcodes from an encoded 2D block in a first frame and an encoded 2D block in a subsequent frame. The collection of Qcodes corresponding to a single encoded 2D block are referred to as an ADRC tile. Accordingly, a motion block generates two ADRC tiles. However, due to the lack of motion, a stationary block need only include one-half of the number of Qcodes of a motion block, thus generating only one ADRC tile. In the present embodiment, the Qcodes of a stationary block are generated by averaging corresponding pixels values between a 2D block in a first frame and a corresponding 2D block in a subsequent frame. Each averaged pixel value is subsequently encoded resulting in the collection of Qcodes forming a single ADRC tile. Accordingly, Motion Blocks 1110 and 1130 generate ADRC Tiles 0, 1, 3, and 4. Stationary Block 1120 generates ADRC Tile 2.

The concatenated ADRC tile generation of Figure 11a concatenates the Qcodes for ADRC Tiles 0 - 4 into Concatenated ADRC Tile B. Specifically, the concatenation is performed for each Qcode (q_0 - q_{63}) included in ADRC Tiles 0, 1, 2, 3 and 4 resulting in the sixty four Qcodes of Concatenated ADRC Tile B.

Alternatively, the generation of each Qcode, Q_i, in Concatenated ADRC Tile B is described by the mathematical equation

$$Q_i = [\, q_{i,0} , q_{i,1} , q_{i,2} , q_{i,3} , q_{i,4} \,] \qquad i = 0, 1, 2, \ldots 63$$

2. Qcode reassignment

Qcode reassignment ensures that bit errors caused by transmission losses are localized within spatially disjointed pixels. In particular, during Qcode reassignment, Qcodes are redistributed and the bits of the redistributed Qcodes are shuffled. Accordingly, Qcode reassignment facilitates error recovery because undamaged pixels surround each damaged pixel. Furthermore, DR and MIN recovery is aided because pixel damage is distributed evenly throughout an ADRC block.

Figure 12 illustrates one embodiment of pixel corruption during the transmission loss of a 1/6 burst error loss. In particular, 2D ADRC Blocks 1210, 1220, and 1230 each include sixty four pixels encoded using three bits. Accordingly, each pixel, P_0 through P_{63}, of a 2D ADRC block is represented by three bits. 2D ADRC Block 1210 shows the bit loss pattern, indicated by a darkened square, of bits when the first bit of every six bits are lost. Similarly, the bit loss pattern when the second bit or fourth bit of every six bits are lost are shown in 2D ADRC Blocks 1220 and 1230, respectively. Figure 12 illustrates that without Qcode reassignment one-half of all the pixels 2D ADRC Blocks 1210, 1220, and 1230 are corrupted for a 1/6 burst error loss.

For one embodiment, Qcode reassignment is applied independently to each concatenated ADRC tile stored in a buffer, thus ensuring that bit errors are localized within spatially disjointed pixels upon deshuffling. In an alternative embodiment, Qcode reassignment is applied to each ADRC block stored in a buffer.

Figure 12a illustrates one embodiment of Qcode reassignment that generates a bitstream of shuffled Qcode bits from a concatenated ADRC tile. Table 122 and Table 132 illustrate the Qcode redistribution. Bitstreams 130 and 140 illustrate the shuffling of Qcode bits.

Table 122 shows the concatenated Qcodes for Concatenated ADRC Tile A. Q_0 is the first concatenated Qcode and Q_{63} is the final concatenated Qcode. Table 132 illustrates the redistribution of Qcodes. For one embodiment Q_0, Q_6, Q_{12}, Q_{18}, Q_{24}, Q_{30}, Q_{36}, Q_{42}, Q_{48}, Q_{54}, and Q_{60} are included in a first set, partition 0. Following Table 132, the following eleven concatenated Qcodes are included in partition 1. The steps are repeated for partitions 2 - 5. The boundary of a partition is delineated by a vertical line in Table 132. This disjointed spatial assignment of concatenated Qcodes to six partitions ensures that a 1/6 burst error loss results in a bit loss pattern distributed across a group of consecutive pixels.

Figure 12b illustrates one embodiment of the bit pattern loss created by the 1/6 burst error loss of redistributed Qcodes. In particular, 2D ADRC blocks 1215, 1225, and 1235 each include sixty four pixels encoded using three bits. Accordingly, each pixel P_0 through P_{63}, of each 2D ADRC block, is represented by three bits. In 2D ADRC Blocks 1215, 1225, and 1235 the bit loss pattern, indicated by a darkened square, is localized across a group of consecutive pixels. Accordingly, only eleven consecutive pixels within each 2D ADRC Block 1215, 1225, and 1235 are corrupted for a given segment loss. In an alternative embodiment, Qcode assignment to partitions include Qcodes from different motion blocks, thus providing both a disjointed spatial and temporal assignment of Qcodes to six segments. This results in additional undamaged spatial-temporal pixels during a 1/6 burst error loss and further facilitates a more robust error recovery.

Referring to **Figure 12a**, the bits of the redistributed Qcodes in Table 132 are shuffled across a generated bitstream so that adjacent bits in the bitstream are from adjacent partitions. The Qcode bits for all the partitions in Table 132 are concatenated into Bitstream 130. For a given partition adjacent bits in Bitstream 130 are scattered to every sixth bit location in the generated Bitstream 140. Accordingly, bits number zero through five, of Bitstream 140, include the first bit from the first Qcode in each partition. Similarly, bits number six through

eleven, of Bitstream 140, include the second bit from the first Qcode in each partition. The process is repeated for all Qcode bits. Accordingly, a 1/6 burst error loss will result in a spatially disjointed pixel loss.

Figure 12c illustrates one embodiment of the bit pattern loss created by the 1/6 burst error loss of reassigned (i.e. redistributed and shuffled) Qcodes. In particular, 2D ADRC Blocks 1217, 1227, and 1237 each include sixty four pixels encoded using three bits. Accordingly, each pixel P_0 through P_{63}, of each 2D ADRC Block, is represented by three bits. In 2D ADRC Blocks 1217, 1227, and 1237, the bit loss pattern, indicated by a darkened square, is distributed across spatially disjointed pixels, thus facilitating pixel error recovery.

3. Randomization of Qcodes

Randomization or shuffling may be applied to destroy the correlation of incorrect candidate decodings that may be generated during a subsequent data decoding process in order to estimate lost or damaged data. The randomization process does not change the properties of the correct candidate decoding, as it is restored to its original condition. In particular, by utilizing randomization, subsequent derandomized data will tend to result in candidate decodings that exhibit highly correlated properties indicative that the corresponding candidate decoding is a good selection.

The randomization process is chosen such that a correct derandomization results in a candidate decoding exhibiting highly correlated properties and an incorrect derandomization results in a decoding exhibiting uncorrelated properties. Encoding parameters such as may be used to perform the randomization and derandomization processes. For example, a randomization pattern may be chosen based on the values of the compression parameters.

One embodiment of a randomization process is illustrated in **Figure 12d**. At step 1277, a bit reallocation is performed. At step 1279 a code reallocation is performed. As noted above, steps 1277 and 1279 each may be performed independently and still realize some coding benefits. In addition, steps 1277 and 1279 may be executed in an order different than illustrated in **Figure 12d**.

In one embodiment, as discussed above, randomization is achieved using a code reallocation process. In one embodiment, reallocation is performed using a masking key. Thus, during the encoding process, a key, referred to herein as KEY, is used to mask a bitstream of Qcodes. KEY may be used to mask a bitstream of Qcodes corresponding to multiple, e.g., three blocks, of data. Each key element (d_i) of the masking key is generated by the combination of one or more compression parameters used to encode a corresponding block of data. This process may enhance error localization.

In one embodiment, the masking process to perform code reallocation results in a randomization of the locations or randomized address mapping of Qcodes across blocks.

The KEY may be generated a variety of ways. In one embodiment, the motion flag (MF) and Qbit values are used to define KEY. Alternately, the masking key may be generated using one or more of the following values: MF, Qbit, DR, CEN, MIN, MAX and the block address of the data of the block.

More particularly, in one embodiment in which 4 bit ADRC encoding is utilized, MF and Qbit values are used to generate KEY. Key may be viewed as a pseudorandom sequence upon which the shuffling process is based. The value of the key elements composing KEY may be determined in accordance with the following equation:

$$KEY = \sum_{i=0}^{N-1} 10^i \cdot d_i$$

where N is the number of blocks of data used, and d_i represents a sub-mask value generated using predetermined parameters such as compression parameters.

Continuing with the present example, if KEY is generated using multiple blocks, e.g., three blocks, KEY is formed according to the following:

$KEY = d_0 + 10 \bullet d_1 + 100 \bullet d_2$

In one embodiment, KEY functions as a mask to indicate locations the Qcodes are to be shuffled to. The result of the process is a randomization of Qcodes, for example, as shown in Figure 12e.

The sub-mask may be determined using a different parameters. In one embodiment, the sub-mask may be defined as:

$$d_i = 5 \bullet m_i + q_i,$$

where q_i represents the number of quantization bits; for example, $q_i = 0, 1, 2, 3, 4,$ and m_i represents the motion flag (MF) value, for example, 0 for a stationary block and 1 for a motion block.

In an alternate embodiment, the sub-mask may be determined based upon a variety of parameters, including, but not limited to, DR, MIN, MAX, CEN, Qbit, motion flag and the block address of the data. For example, if DR, the Qbit value and the motion flag are used, the sub-mask may be determined as:

$$d_i = (10 \bullet DR_i) + (5 \bullet m_i) + q_i$$

If the block address (BA), the Qbit value and the motion flag are used, the sub-mask may be determined as:

$$d_i = (10 \bullet BA_i) + (5 \bullet m_i) + q_i$$

If the DR value, the block address, the Qbit value and the motion flag are used, the sub-mask may be determined as:

$$d_i = (2560 \bullet BA_i) + (10 \bullet DR_i) + (5 \bullet m_i) + q_i$$

If recovery of certain parameters is required, for example, MF or Qbit data, a derandomization process is performed in which possible KEY values are regenerated depending upon the values used to create the masking keys. The regenerated KEY values are used to unmask the received bitstream of Qcodes resulting in candidate encoded data. Thus, if the MF or Qbit value used to generate the mask is not correct, the corresponding Qcodes will exhibit a low level of correlation, which will be typically readily detectable.

In another embodiment, a randomization process, referred to herein as bit reallocation, is applied to the data. In one embodiment, bit reallocation is achieved by simple bit weight inversion. The inversion pattern may be

determined according to the number of bits used for encoding (e.g., Qbit). This randomization process can improve recovery of MF and Qbit values. Examples are shown in **Figures 12e, 12f, 12g, and 12h**. **Figure 12f** illustrates a bit reallocation process for 2 bit encoding, **Figure 12g** illustrates a bit reallocation for 3 bit encoding and **Figure 12h** illustrates a bit reallocation for 4 bit encoding.

Alternate processes may be applied to perform code reallocation and/or bit reallocation. For example, bit reallocation may be dependent upon one or more parameters such as is discussed above with respect to code reallocation. Weight inversion processes may also be applied to code reallocation.

In an alternate embodiment, a compression parameter such as the Qbit value of a block may be used as a randomizing or seed value for a pseudorandom number generator (PNG). In one embodiment, the PNG may create a statistically distinct pseudorandom number sequence for each unique seed value as well as a corresponding or the same statistically distinct sequence for each application of the same seed value.

In one embodiment, the pseudorandom sequence may be used to transform the VL-data on a bit by bit or code by code basis. In alternate embodiments, the FL-data may be transformed or both the VL-data and FL-data may be transformed.

In one embodiment, the transformation T of the VL-data may be achieved by applying a bitwise XOR (exclusive OR) function to the pseudorandom number sequence (y) and the VL-data (x). Thus:

$$T(x) = x \oplus y.$$

In this embodiment, the bitwise XOR function is used, as the inverse transformation is exactly the same as for the original, forward transformation. That is:

$$T^{-1}(T(x)) = (x \oplus y) \oplus y = x.$$

In alternate embodiments, a variety of sets of transformations may be used to generate the statistically distinct sequences. For example, a table of pre-defined sequences may be used.

The seed value may be based upon selected parameters such as compression parameters. In one embodiment, the Qbit value of the block is used as the seed value. Other values based upon DR, MF, CEN, MIN, MAX and block address may also be used. For example, the seed value may be determined as $(5 \bullet m_i + q_i).(10 \bullet DR_i) + (5 \bullet m_i) + q_i$, or $(2560 \bullet BA_i) + (10 \bullet DR_i) + (5 \bullet m_i) + q_i$.

In one embodiment, a similar process may be used to decode the parameter. For example, a similar process may be used to decode the Qbit value from the DR of the current block. If the DR arrives undamaged, the Qbit value may be determined by using the threshold table as was used for the Qcode encoding. The DR is used to look-up the Qbit value in the table and the Qbit value is then used as a seed value to the PNG to produce the pseudorandom number sequence. The decoder transforms the randomized VL-data by applying a bitwise XOR function to the pseudorandom number sequence and the randomized VL-data to produce the original, non-randomized VL-data. In this embodiment, because the same PNG and seed value are used, the same pseudorandom number sequence is produced. In alternate embodiments, corresponding variations of the PNG and seed value may be used and corresponding process steps may be applied to determine a pseudorandom sequence.

In one embodiment, if the DR is damaged or lost, the decoder may attempt to decode the block with all possible Qbit values and associated possible seed values. A local correlation metric is applied to each candidate decoding, and a confidence metric is computed for the block.

Thus, in one embodiment, enhanced recovery of a lost or damaged Qbit value may is realized by means of shuffling of Qcodes. In alternate embodiments, the shuffling process described can utilize a variety of types of data to enchance subsequent recovery of that data if lost or damaged. For example, the Motion Flag, DR or block address of the data, or a combination of the Qbit value, the Motion Flag, DR and/or block address may be used to generate the seed value.

Figures 10 - 12 illustrate intra group VL-data shuffling tolerated up to 1/6 packet data loss during transmission. It will be appreciated by one skilled in the art, that the number of total partitions and bit separation can be varied to ensure against 1/n burst error loss.

Inter Segment FL-Data Shuffling

Inter segment FL-data shuffling describes rearranging block attributes among different segments. Rearranging block attributes provides for a distributed loss of data. In particular, when FL-data from a segment is lost during transmission the DR value, MIN value, and Motion Flag value lost do not belong to the same block. **Figures 13 and 14** illustrate one embodiment of inter segment FL-data shuffling.

Figure 13 illustrates the contents of Segments 0 to 5. For one embodiment, each segment comprises 880 DRs, 880 MINs, 880 Motion Flags, and VL-data corresponding to 660 Y-blocks, 110 U-blocks, and 110 V-blocks. As illustrated in graph MIN Shuffling 1300, the MIN values for Segment 0 are moved to Segment 2, the MIN values for Segment 2 are moved to Segment 4, and the MIN values for Segment 4 are moved to Segment 0. Additionally, the MIN values for Segment 1 are moved to Segment 3, the MIN values for Segment 3 are moved to Segment 5, and the Motion Flag values for Segment 5 are moved to Segment 1.

Figure 13a illustrates Motion Flag shuffling. As illustrated, in graph Motion Flag Shuffling 1305, the Motion Flag values for Segment 0 are moved to Segment 4, the Motion Flag values for Segment 2 are moved to Segment 0, and the Motion Flag values for Segment 4 are moved to Segment 2. Additionally, the Motion Flag values for Segment 1 are moved to Segment 5, the Motion Flag values for Segment 3 are moved to Segment 1, and the Motion Flag values for Segment 5 are moved to Segment 3. Loss pattern 1310 illustrates the FL-data loss after Segment 0 is lost during transmission.

For a specific block attribute, both **Figure 13** and **Figure 13a** illustrate shuffling all instances of the specific block attribute between segments. For example, in **Figure 13** the 880 MIN values from Segment 0 are collectively

exchanged with the 880 MIN values in Segment 2. Similarly, in **Figure 13a** the 880 Motion Flags for Segment 0 are collectively exchanged with the 880 Motion Flags in Segment 4. During a transmission loss of consecutive packets, this collective shuffling of block attributes results in a disproportionate loss of a specific block attributes for a block group. In one embodiment, a block group includes three ADRC blocks.

Figure 14 illustrates one embodiment of a modular three shuffling process for DR, MIN, and Motion Flag values. A modular three shuffling describes a shuffling pattern shared across three blocks (i.e., a block group) in three different segments. The shuffling pattern is repeated for all block groups within the three different segments. However, a different shuffling pattern is used for different block attributes. Accordingly, the modular three shuffling process distributes block attributes over all three segments. In particular, for a given block group a modular three shuffling ensures that only one instance of a specific block attribute is lost during the transmission loss of a segment. Thus, during the data recovery process, described below, a reduced number of candidate decodings are generated to recover data loss within a block.

As illustrated in DR Modular Shuffle 1410, a segment stores 880 DR values. Accordingly, the DR values are numbered 0 - 879 dependent on the block from which a given DR value is derived. In a modular three shuffling the FL-data contents of three segments are shuffled. A count of 0 - 2 is used to identify each DR value in the three segments identified for a modular shuffling. Accordingly, DR's belonging to blocks numbered 0, 3, 6, 9 . . . belong to Count 0. Similarly, DR's belonging to blocks numbered 1, 4, 7, 10, . . . belong to Count 1 and DR's belonging to blocks numbered 2, 5, 8, 11 . . . belong to Count 2. Thus, for a given count the DR values associated with that count are shuffled across Segment 0, 2, and 4. Similarly, the DR values associated with the same count are shuffled across Segments 1, 3, and 5.

In DR Modular Shuffle 1410, the DR values belonging to Count 0 are left un-shuffled. The DR values belonging to Count 1 are shuffled. In particular, the Count 1 DR values in Segment A are moved to Segment B, the Count 1 DR

values in Segment B are moved to Segment C, and the Count 1 DR values in Segment C are moved to Segment A.

The DR values belonging to Count 2 are also shuffled. In particular, the Count 2 DR values in Segment A are moved to Segment C, the Count 2 DR values in Segment B are moved to Segment A, and the Count 2 DR values in Segment C are moved to Segment B.

MIN Modular Shuffle 1420 illustrates one embodiment of a modular three block attribute shuffling process for MIN values. A segment includes 880 MIN values. In MIN Modular Shuffle 1420, the shuffling pattern used for Count 1 and Count 2 in DR Modular Shuffle 1410 are shifted to Count 0 and Count 1. In particular, the shuffling pattern used for Count 1 in DR Modular Shuffle 1410 is applied to Count 0. The shuffling pattern used for Count 2 in DR Modular Shuffle 1410 is applied to Count 1 and the MIN values belonging to Count 2 are left un-shuffled.

Motion Flag Modular Shuffle 1430 illustrates one embodiment of a modular three block attribute shuffling process for Motion Flag values. A segment includes 880 Motion Flag values. In Motion Flag Modular Shuffle 1430, the shuffling pattern used for Count 1 and Count 2 in DR Modular Shuffle 1410 are shifted to Count 2 and Count 0 respectively. In particular, the shuffling pattern used for Count 2 in DR Modular Shuffle 1410 is applied to Count 0. The shuffling pattern used for Count 1 in DR Modular Shuffle 1410 is applied to Count 2 and the Motion Flag values belonging to Count 1 are left un-shuffled.

Figure 14a illustrates the modular shuffling result of Modular Shuffles 1410, 1420, and 1430. Modular Shuffle Result 1416 shows each attribute destination of blocks belonging to Segment 0. In this example, Segment 0 corresponds to Segment A of **Figure 14**. This destination is defined according to Modular Shuffles 1410, 1420, and 1430 of **Figure 14**. **Figure 14a** also illustrates the distribution loss of block attributes after Segment 0 is lost during transmission. In particular, Loss Pattern 1415 shows the DR, Motion Flag, and MIN values loss across six segments after a subsequent deshuffling is applied to the received data that was initially shuffled using Modular Shuffles 1410, 1420,

and 1430. As illustrated in **Figure 14a**, the block attribute loss is distributed periodically across Segments 0, 2, and 4 while Segments 1, 3, and 5 have no block attribute loss. Additionally, Spatial Loss Pattern 1417 illustrates the deshuffled spatial distribution of damaged FL-data after Segment 0 is lost during transmission. Spatial Loss Pattern 1417 shows the DR, Motion Flag, and MIN value loss after a subsequent deshuffling is applied to the received data. In Spatial Loss Pattern 1417, a damaged block is surrounded by undamaged blocks and damaged block attributes can be recovered with surrounding undamaged blocks.

Figure 14 and **Figure 14a** illustrate a modular three shuffling pattern and the distribution loss of block attributes after a segment is lost during transmission. In alternative embodiments, the count variables or the number of segments are varied to alternate the distribution of lost block attributes. **Figure 14b** illustrates Modular Shuffle Result 1421 and Loss Pattern 1420. Similarly, **Figure 14c** illustrates Modular Shuffle Result 1426 and Loss Pattern 1425. Both Loss Pattern 1420 and Loss Pattern 1425 illustrate the distribution loss of block attributes across six segments, as opposed to three segments as previously described.

It is contemplated that in alternate embodiments various combinations of block attributes will be distributed to perform the shuffling process.

Inter Segment VL-Data Shuffling

In the inter segment VL-data shuffling process, bits between a predetermined number of segments, for example, 6 segments, are arranged to ensure a spatially separated and periodic VL-data loss during an up to 1/6 packet transmission loss. **Figure 15** and **16** illustrate one embodiment of the inter segment VL-data shuffling process.

In the present embodiment, a transmission rate approaching 30 Mbps is desired. Accordingly, the desired transmission rate results in 31,152 bits available for the VL-data in each of the 60 buffers. The remaining space is used by FL-data for the eighty eight blocks included in a buffer. **Figure 15** includes the VL-data buffer organization within a frame set for a transmission rate

approaching 30 Mbps. As previously described, partial buffering is used to maximize the usage of available VL-data space within each buffer, and the unused VL-data space is filled with a post-amble.

Figure 16 illustrates one embodiment of the shuffling process to ensure a spatially separated and periodic VL-data loss. The first row illustrates the VL-data from the 60 buffers in Figure 15 rearranged into a concatenated stream of 1,869,120 bits. The second row illustrates the collection of every sixth bit into a new stream of bits. Thus, when the decoder subsequently reverses the process, a burst loss of up to 1/6 of the data transmitted is transformed into a periodic loss where at least 5 undamaged bits separate every set of two damaged bits.

The third row illustrates grouping every 10 bits of Stream 2 into a new stream of bits, Stream 3. The boundary of a grouping is also defined by the number of bits in a segment. Grouping of Stream 2 for every tenth bit ensures that a 1/60 data loss results in fifty-nine undamaged bits between every set of two damaged bits. This provides for a spatially separated and periodic VL-data loss in the event that 88 consecutive packets of data are lost.

The fourth row illustrates grouping every 11 bits of Stream 3 into Stream 4. The boundary of a grouping is also defined by the number of bits in a segment. Grouping of Stream 3 for every eleventh bit ensures that 1/660 data loss results in 659 undamaged bits between to damaged bits, resulting in a spatially separated and periodic VL-data loss during a transmission loss of 8 consecutive packets.

Each group of 31,152 bits within Stream 4 is consecutively re-stored in Buffers 0 - 59, with the first group of bits stored in Buffer 0 and the last group of bits stored in Buffer 59.

It will be appreciated by one skilled in the art that the grouping requirements of **Figure 16** are variable to ensure a spatially separated and periodic VL-data loss tolerance up to a 1/n transmission loss.

Transmission

The previously described shuffling process creates buffers with intermixed FL-data and VL-data. For one embodiment, packets are generated from each buffer, according to packet structure 200, and transmitted across Transmission media 135. The data received is subsequently decoded. Lost or damaged data may be recovered using data recovery processes.

Decoding

Referring again to **Figure 4**, a flow diagram illustrating one embodiment of a decoding process performed by decoder 120 is shown. In one embodiment, the conversion and de-shuffling processes are the inverse of the processes represented in Figure 3. These processes include the code reallocation and bit reallocation processes discussed with reference to **Figures 12d-12h**.

The invention has been described in conjunction with the preferred embodiment. It is evident that numerous alternatives, modifications, variations and uses will be apparent to those skilled in the art in light of the foregoing description.

CLAIMS

What is claimed is:

1. A method for encoding data to provide for recovery of lost or damaged encoded data during subsequent decoding, said method comprising:

defining a pseudorandom sequence based upon at least one compression constant representative of the encoded data; and

shuffling the data using the pseudorandom sequence.

2. The method as set forth in claim 1, wherein the shuffling comprises code reallocation based upon the pseudorandom sequence.

3. The method as set forth in claim 2, wherein code reallocation comprises address remapping based upon the pseudorandom sequence.

4. The method as set forth in claim 1, wherein the shuffling comprises bit reallocation based upon the pseudorandom sequence.

5. The method as set forth in claim 4, wherein bit reallocation comprises exclusive- OR ing the pseudorandom sequence with the encoded data.

6. The method as set forth in claim 2, wherein the data is encoded using Adaptive Dynamic Range Coding (ADRC), said code reallocation shuffling locations of Qcodes based upon the pseudorandom sequence.

7. The method as set forth in claim 4, wherein the data is encoded using Adaptive Dynamic Range Coding (ADRC), said bit reallocation shuffling locations of bits of Qcodes based upon the pseudorandom sequence.

8. The method as set forth in claim 1, wherein defining a pseudorandom sequence comprises generating a seed value based upon the at least one compression parameter, said pseudorandom sequence based upon the seed value.

9. The method as set forth in claim 8, further comprising using the seed value as input to a pseudorandom number generator, said pseudorandom number generator outputting a pseudorandom sequence.

10. The method as set forth in claim 8, further comprising generating an pseudorandom address mapping based upon the seed value.

11. The method as set forth in claim 8, wherein the seed value is based upon a plurality of values selected from the group comprising a dynamic range value, block address, number of quantization bits and a motion flag value.

12. The method as set forth in claim 8, wherein the data is encoded using Adaptive Dynamic Range Coding (ADRC), and the seed value is generated according to an equation selected from the group comprising $(5 \bullet m_i + q_i)$, $(10 \bullet DR_i) + (5 \bullet m_i) + q_i$, and $(2560 \bullet BA_i) + (10 \bullet DR_i) + (5 \bullet m_i) + q_i$, where i represents an ith block, m_i represent the motion flag, q_i represents the Qbit value, DR_i represents a dynamic range of data and BA_i represents a block address of the data.

13. The method as set forth in claim 1, wherein the data is encoded using Adaptive Dynamic Range Coding (ADRC), and the pseudorandom sequence is generated based upon a value determined according to an equation selected from the group comprising $(5 \bullet m_i + q_i)$, $(10 \bullet DR_i) + (5 \bullet m_i) + q_i$, and $(2560 \bullet BA_i) + (10 \bullet DR_i) + (5 \bullet m_i) + q_i$, where i represents an ith block, m_i represent the motion flag, q_i represents the Qbit value, DR_i represents a dynamic range of data and BA_i represents a block address of the data.

14. The method of claim 1 wherein data is selected from the group consisting of two-dimensional static images, hologram images, three-dimensional static images, video, two-dimensional moving images, three dimensional moving images, monaural sound, and N-channel sound.

15. A system for encoding data to provide for recovery of lost or damaged encoded data during subsequent decoding, said method comprising:

a pseudorandom sequence based upon at least one compression constant representative of the encoded data; and

shuffling logic configured to shuffle the data using the pseudorandom sequence.

16. The system as set forth in claim 15, wherein the shuffle comprises code reallocation based upon the pseudorandom sequence.

17. The system as set forth in claim 16, wherein code reallocation comprises address remapping based upon the pseudorandom sequence.

18. The system as set forth in claim 15, wherein the shuffle comprises bit reallocation based upon the pseudorandom sequence.

19. The system as set forth in claim 18, wherein bit reallocation comprises exclusive- OR ing the pseudorandom sequence with the encoded data.

20. The system as set forth in claim 16, wherein the data is encoded using Adaptive Dynamic Range Coding (ADRC), said code reallocation shuffling locations of Qcodes based upon the pseudorandom sequence.

21. The system as set forth in claim 18, wherein the data is encoded using Adaptive Dynamic Range Coding (ADRC), said bit reallocation shuffling locations of bits of Qcodes based upon the pseudorandom sequence.

22. The system as set forth in claim 15, wherein the pseudorandom sequence is based upon a seed value based upon the at least one compression parameter.

23. The system as set forth in claim 22, further comprising a pseudorandom number generator which uses the seed value as input, said pseudorandom number generator configured to output a pseudorandom sequence.

24. The system as set forth in claim 22, wherein the pseudorandom sequence comprises a pseudorandom address mapping based upon the seed value.

25. The system as set forth in claim 22, wherein the seed value is based upon a plurality of values selected from the group comprising a dynamic range value, block address, number of quantization bits and a motion flag value.

26. The system as set forth in claim 22, wherein the data is encoded using Adaptive Dynamic Range Coding (ADRC), and the seed value is generated according to an equation selected from the group comprising $(5 \bullet m_i + q_i)$, $(10 \bullet DR_i) + (5 \bullet m_i) + q_i$, and $(2560 \bullet BA_i) + (10 \bullet DR_i) + (5 \bullet m_i) + q_i$, where i represents an ith block, m_i represent the motion flag, q_i represents the Qbit value,

DR_i represents a dynamic range of data and BA_i represents a block address of the data.

27. The system as set forth in claim 15, wherein the data is encoded using Adaptive Dynamic Range Coding (ADRC), and the pseudorandom sequence is generated based upon a value determined according to an equation selected from the group comprising $(5 \bullet m_i + q_i)$, $(10 \bullet DR_i) + (5 \bullet m_i) + q_i$, and $(2560 \bullet BA_i) + (10 \bullet DR_i) + (5 \bullet m_i) + q_i$, where i represents an ith block, m_i represent the motion flag, q_i represents the Qbit value, DR_i represents a dynamic range of data and BA_i represents a block address of the data.

28. The system as set forth in claim 15 wherein data is selected from the group consisting of two-dimensional static images, hologram images, three-dimensional static images, video, two-dimensional moving images, three . dimensional moving images, monaural sound, and N-channel sound.

29. The system as set forth in claim 15, wherein the shuffling logic is selected from the group comprising at least one processor, at least one large scale integration (LSI) component and at least one ASIC. 30. A computer readable medium comprising instructions, which when executed on a processor, perform method for encoding data to provide for recovery of lost or damaged encoded data during subsequent decoding, comprising:

defining a pseudorandom sequence based upon at least one compression constant representative of the encoded data; and

shuffling the data using the pseudorandom sequence.

31. The computer readable medium as set forth in claim 30 wherein the shuffling comprises code reallocation based upon the pseudorandom sequence.

32. The computer readable medium as set forth in claim 31 wherein code reallocation comprises address remapping based upon the pseudorandom sequence.

33. The computer readable medium as set forth in claim 30 wherein the shuffling comprises bit reallocation based upon the pseudorandom sequence.

34. The computer readable medium as set forth in claim 33, wherein bit reallocation comprises exclusive- OR ing the pseudorandom sequence with the encoded data.

35. The computer readable medium as set forth in claim 31 wherein the data is encoded using Adaptive Dynamic Range Coding (ADRC), said code reallocation shuffling locations of Qcodes based upon the pseudorandom sequence.

36. The computer readable medium as set forth in claim 33, wherein the data is encoded using Adaptive Dynamic Range Coding (ADRC), said bit reallocation shuffling locations of bits of Qcodes based upon the pseudorandom sequence.

37. The computer readable medium as set forth in claim 30 wherein defining a pseudorandom sequence comprises generating a seed value based upon the at least one compression parameter, said pseudorandom sequence based upon the seed value.

38. The computer readable medium as set forth in claim 37, further comprising instructions, which when executed, use the seed value as input to a pseudorandom number generator, said pseudorandom number generator outputting a pseudorandom sequence.

39. The computer readable medium as set forth in claim 37, further comprising instructions, which when executed, generate a pseudorandom address mapping based upon the seed value.

40. The computer readable medium as set forth in claim 37, wherein the seed value is based upon a plurality of values selected from the group comprising a dynamic range value, block address, number of quantization bits and a motion flag value.

41. The computer readable medium as set forth in claim 37, wherein the data is encoded using Adaptive Dynamic Range Coding (ADRC), and the seed value is generated according to an equation selected from the group comprising $(5 \bullet m_i + q_i)$, $(10 \bullet DR_i) + (5 \bullet m_i) + q_i$, and $(2560 \bullet BA_i) + (10 \bullet DR_i) + (5 \bullet m_i) + q_i$, where i represents an ith block, m_i represent the motion flag, q_i

represents the Qbit value, DR_i represents a dynamic range of data and BA_i represents a block address of the data.

42. The method as set forth in claim 30 wherein the data is encoded using Adaptive Dynamic Range Coding (ADRC), and the pseudorandom sequence is generated based upon a value determined according to an equation selected from the group comprising $(5 \bullet m_i + q_i)$, $(10 \bullet DR_i) + (5 \bullet m_i) + q_i$, and $(2560 \bullet BA_i) + (10 \bullet DR_i) + (5 \bullet m_i) + q_i$, where i represents an ith block, m_i represent the motion flag, q_i represents the Qbit value, DR_i represents a dynamic range of data and BA_i represents a block address of the data.

43. The computer readable medium as set forth in claim 30, wherein data is selected from the group consisting of two-dimensional static images, hologram images, three-dimensional static images, video, two-dimensional moving images, three dimensional moving images, monaural sound, and N-channel sound.

44. An apparatus configured to encode data to provide for recovery of lost or damaged encoded data during subsequent decoding, comprising:

means for defining a pseudorandom sequence based upon at least one compression constant representative of the encoded data; and

mean for shuffling the data using the pseudorandom sequence.

45. The apparatus as set forth in claim 44, wherein the shuffling comprises code reallocation based upon the pseudorandom sequence.

46. The apparatus as set forth in claim 44, further comprising a pseudorandom number generator, said pseudorandom number generator outputting a pseudorandom sequence using the seed value as input.

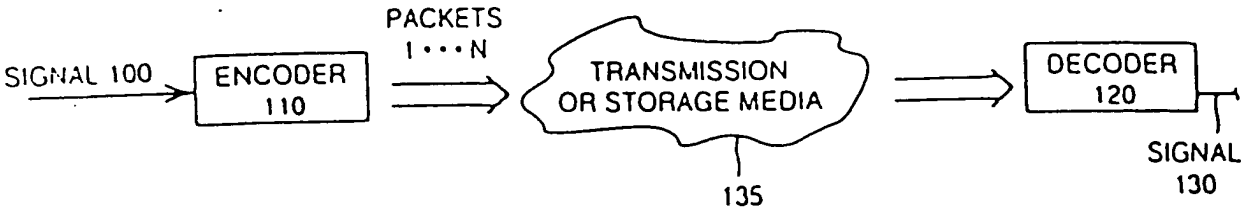


Fig. 1A

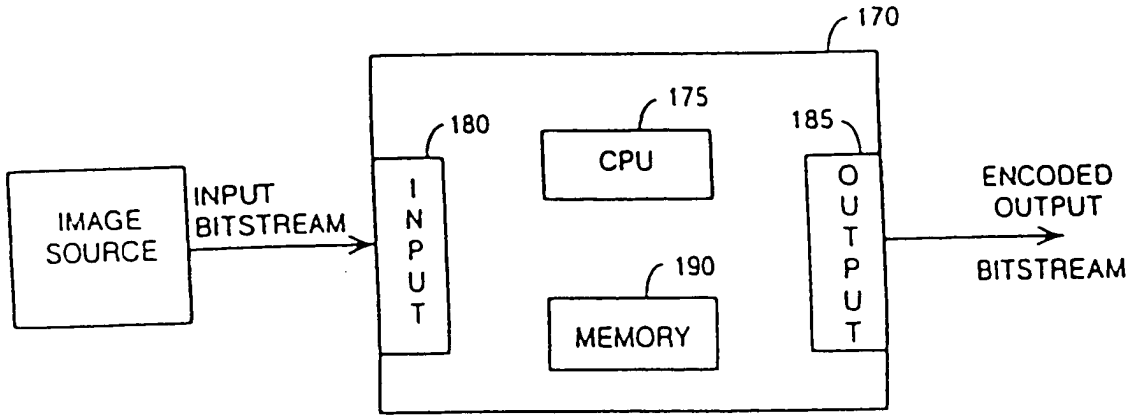


Fig. 1B

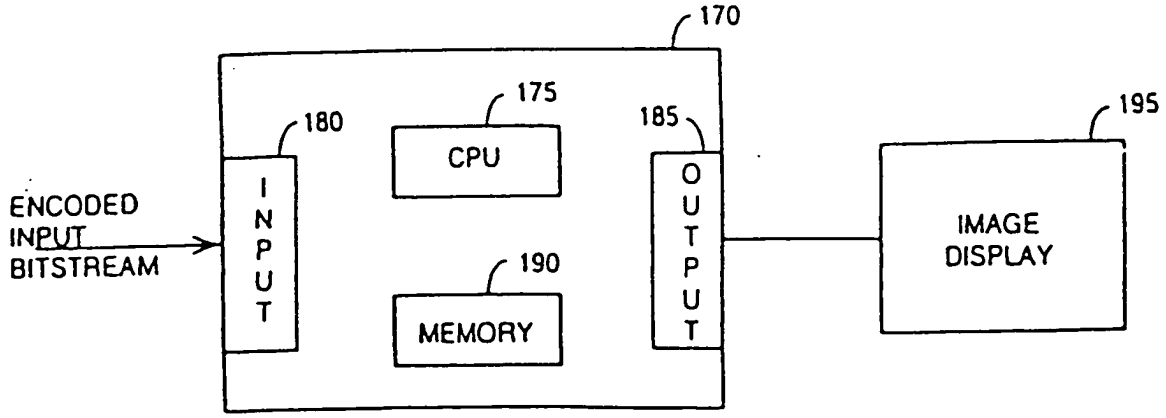


Fig. 1C

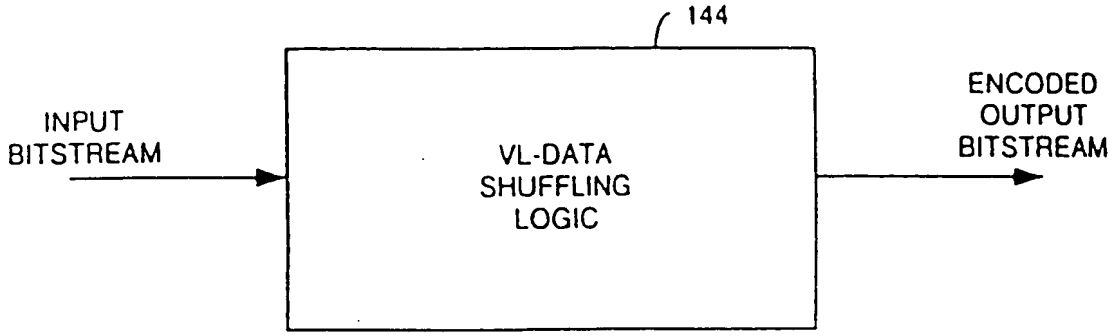


FIG. 1d

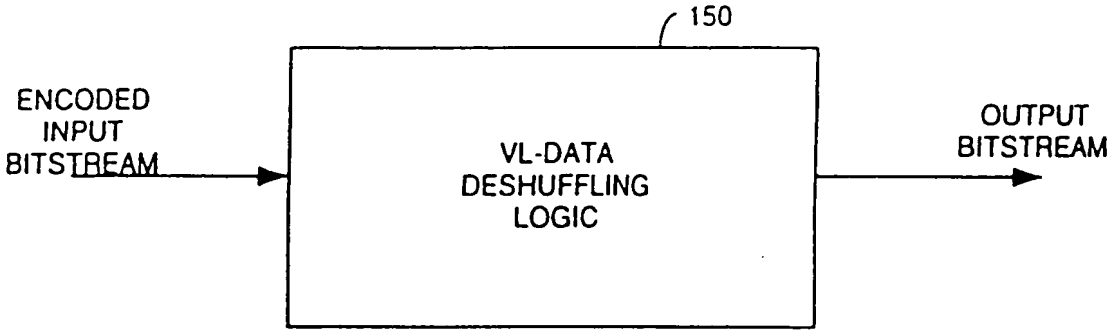


FIG. 1e

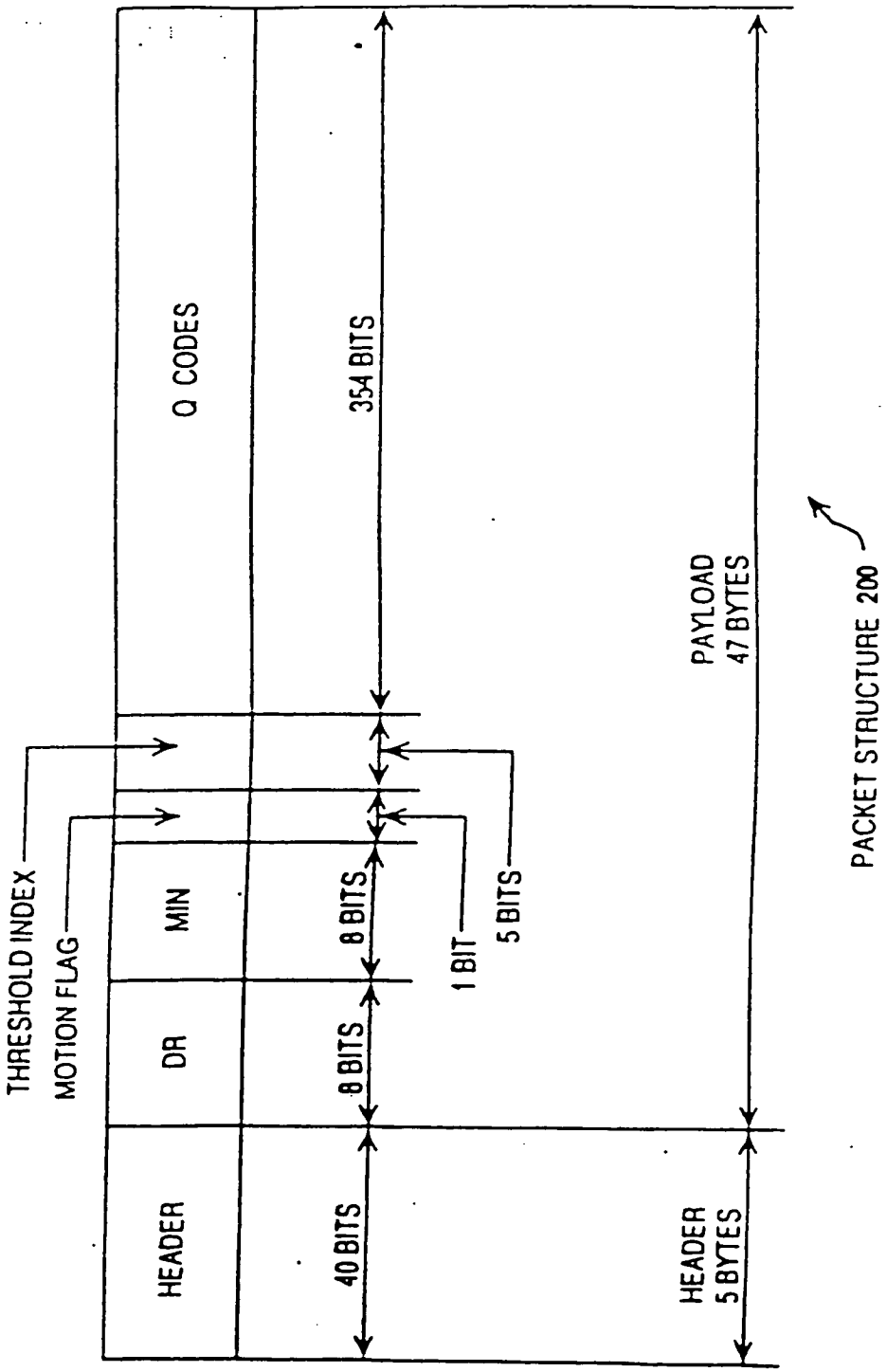
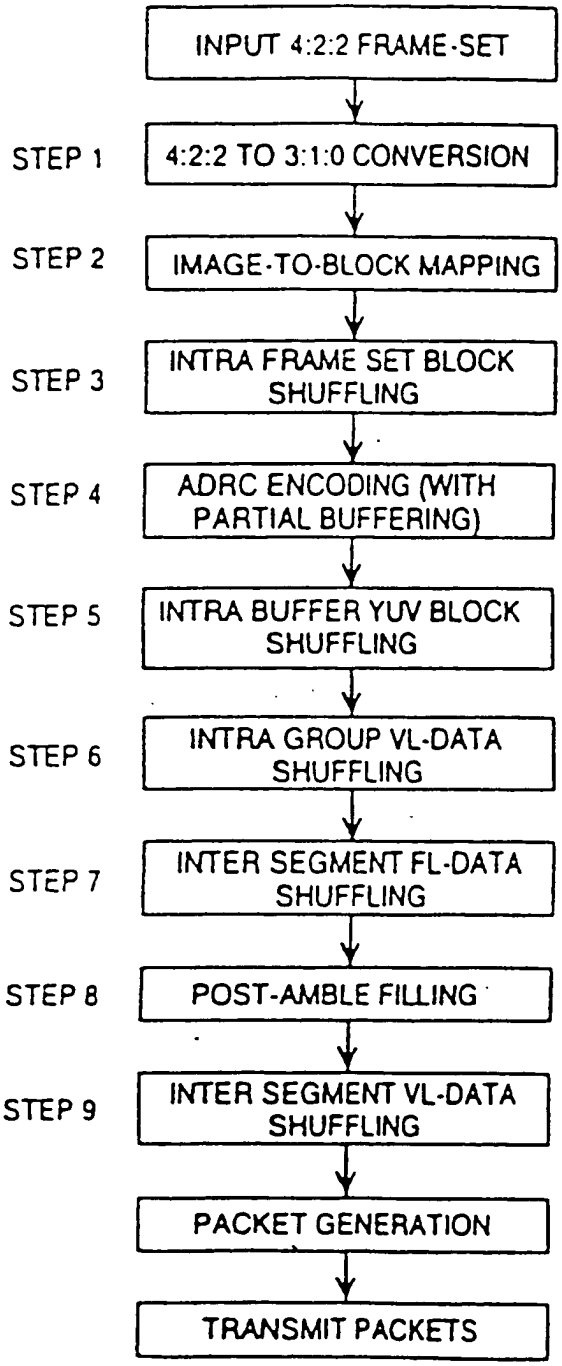

Q CODES
354 BITS
PAYLOAD
47 BYTES
PACKET STRUCTURE 200
THRESHOLD INDEX
MOTION FLAG
MIN
DR
HEADER
8 BITS
8 BITS
1 BIT
5 BITS
40 BITS
HEADER
5 BYTES

FIG. 2



SIGNAL FLOWGRAPH IN THE ENCODER

FIG. 3

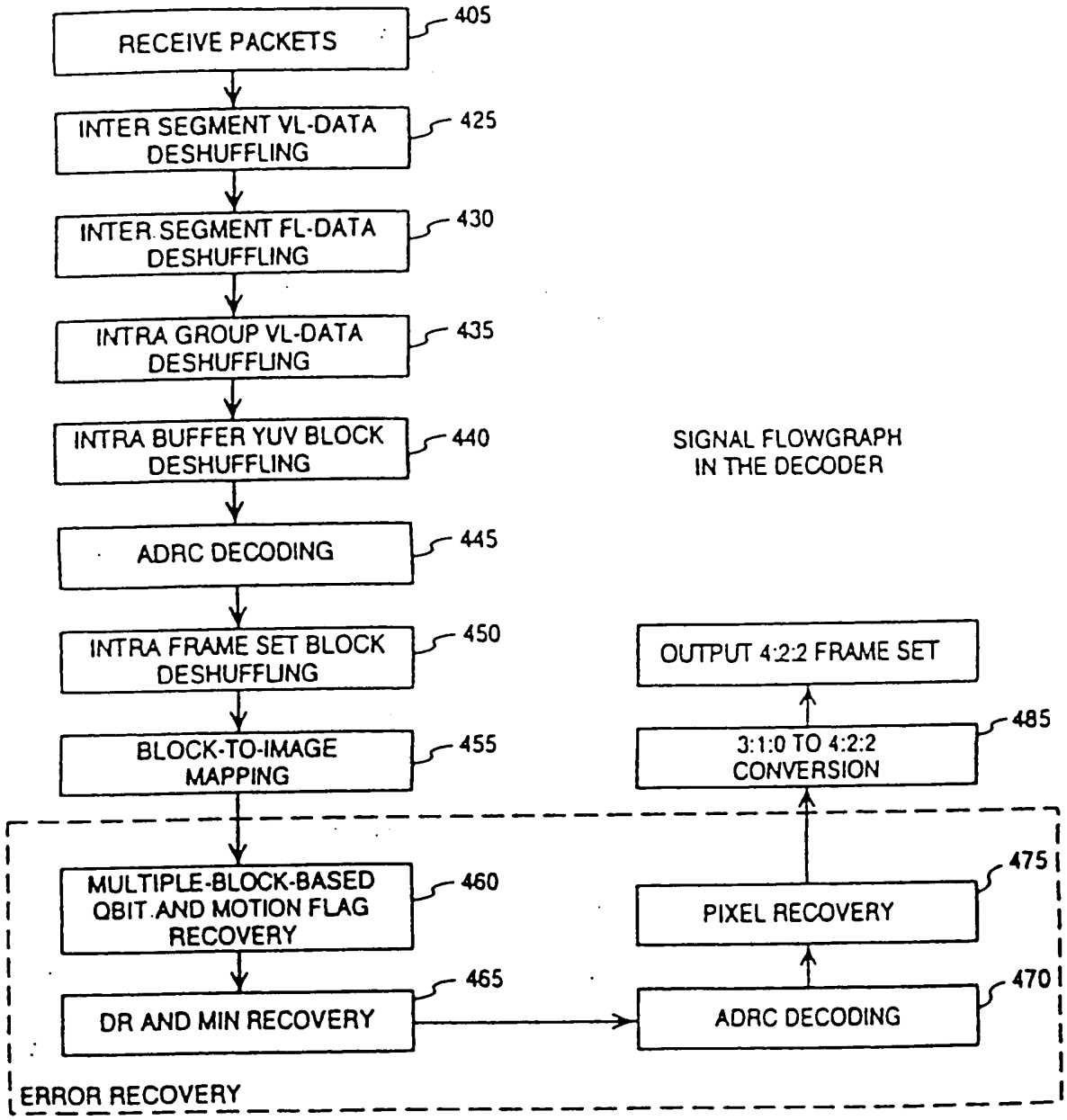

RECEIVE PACKETS
405
INTER SEGMENT VL-DATA DESHUFFLING
425
INTER SEGMENT FL-DATA DESHUFFLING
430
INTRA GROUP VL-DATA DESHUFFLING
435
INTRA BUFFER YUV BLOCK DESHUFFLING
440
ADRC DECODING
445
INTRA FRAME SET BLOCK DESHUFFLING
450
BLOCK-TO-IMAGE MAPPING
455
MULTIPLE-BLOCK-BASED QBIT AND MOTION FLAG RECOVERY
460
DR AND MIN RECOVERY
465
ERROR RECOVERY
SIGNAL FLOWGRAPH IN THE DECODER
OUTPUT 4:2:2 FRAME SET
3:1:0 TO 4:2:2 CONVERSION
485
PIXEL RECOVERY
475
ADRC DECODING
470

FIG. 4

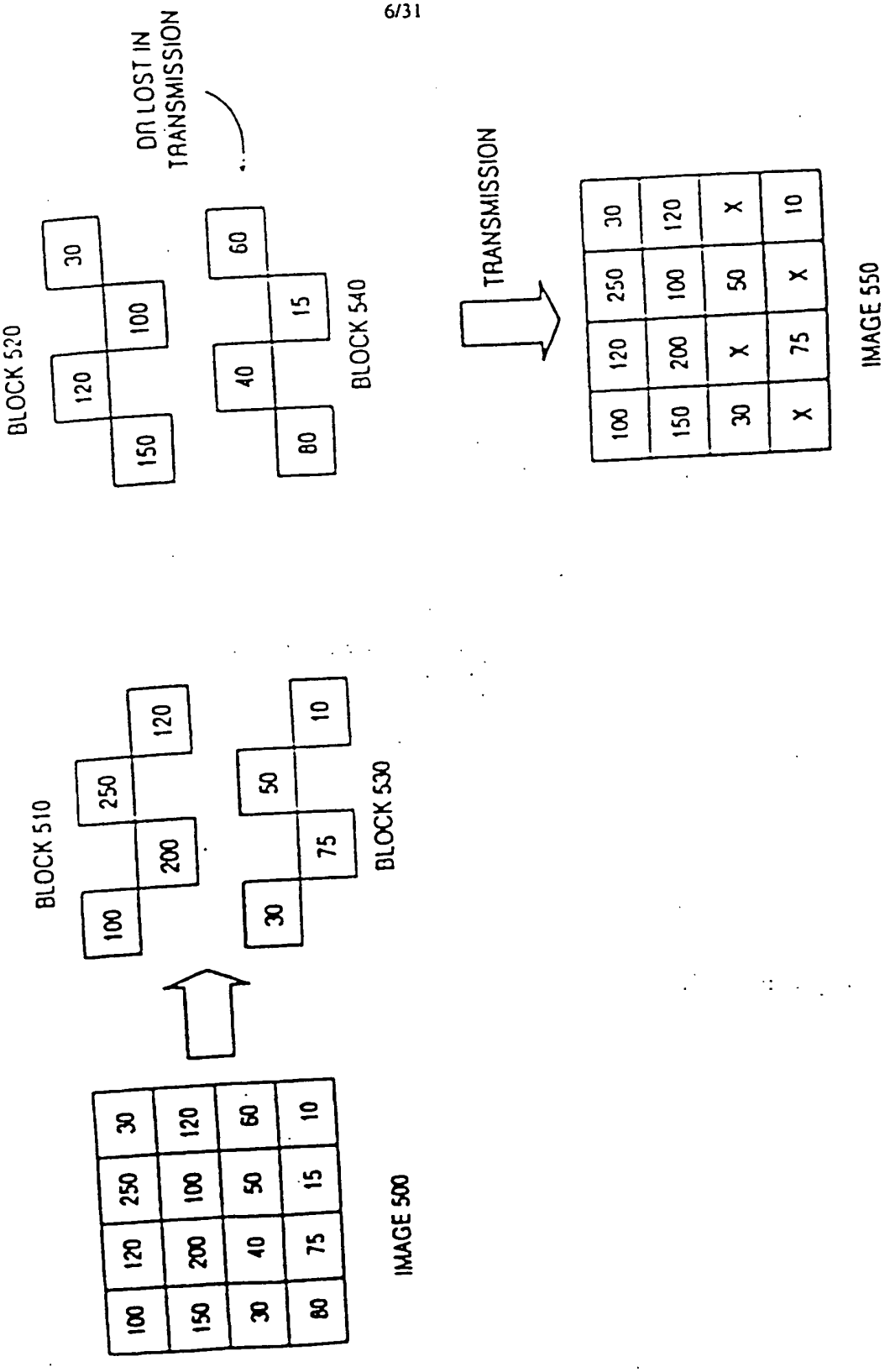
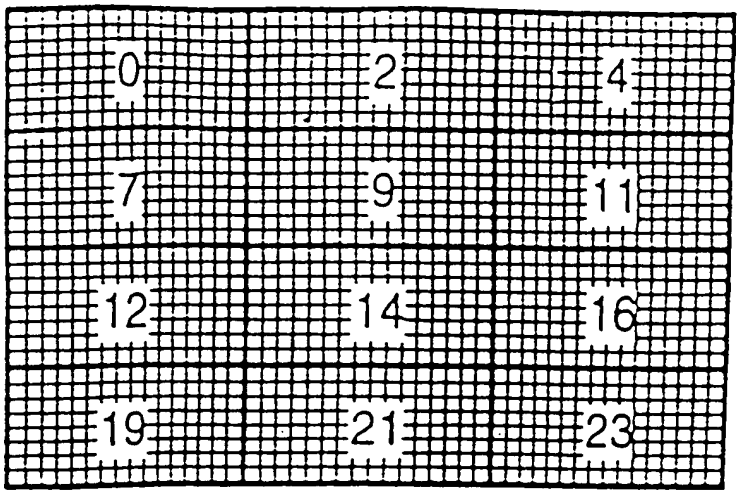
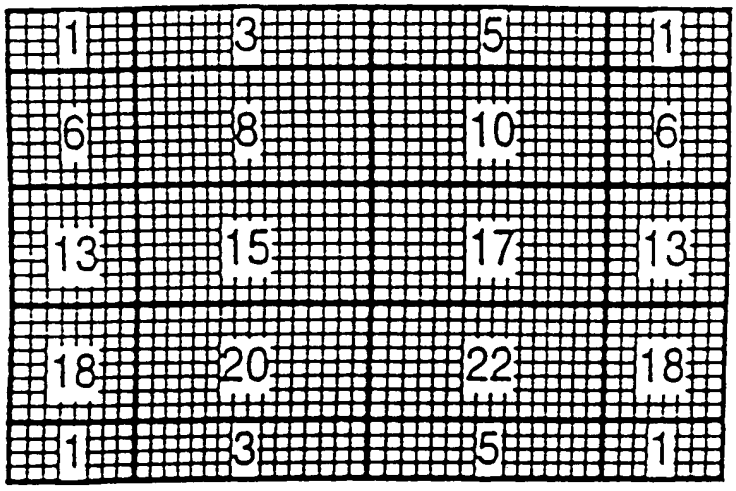


IMAGE 500
100 150 30 80
120 200 40 75
250 100 50 15
30 120 60 10
BLOCK 510
100 150 120 200
250 120
BLOCK 530
30 75
50 10
BLOCK 520
150 120 100
30
BLOCK 540
40 80 15
60
DATA LOST IN TRANSMISSION
TRANSMISSION
IMAGE 550
100 150 120 200
120 250 100
30 X 50
X 75 X
30 X 10

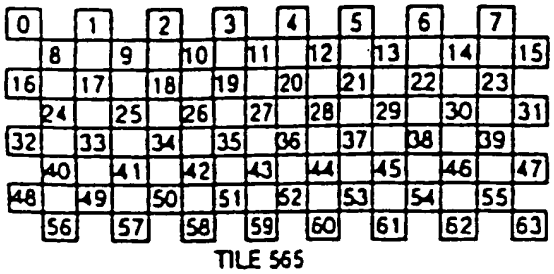
FIG. 5



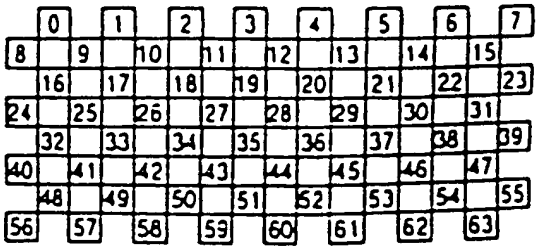
SUB-IMAGE 560



SUB-IMAGE 570

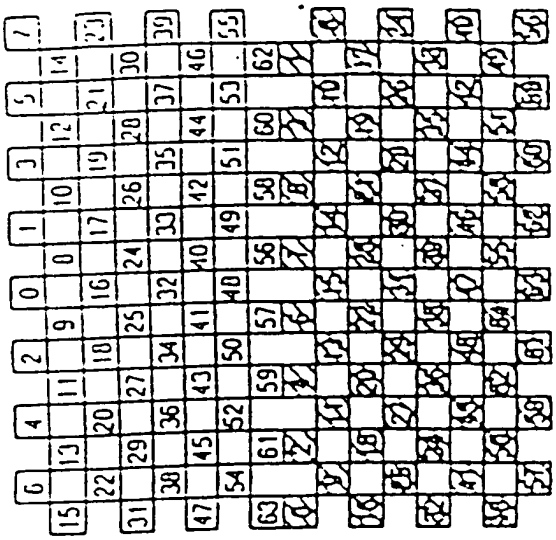


TILE 565

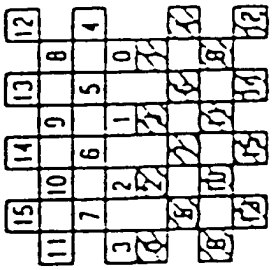


TILE 575

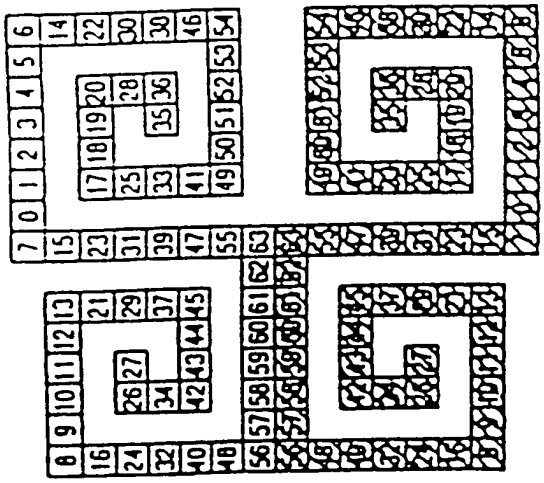
FIG. 5a



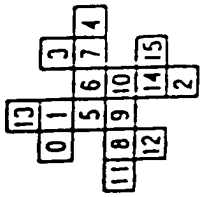
610b



610d



610a



610c

FIG. 6

INTRA FRAME SET BLOCK SHUFFLING
SEGMENT DEFINITION: Y BLOCKS

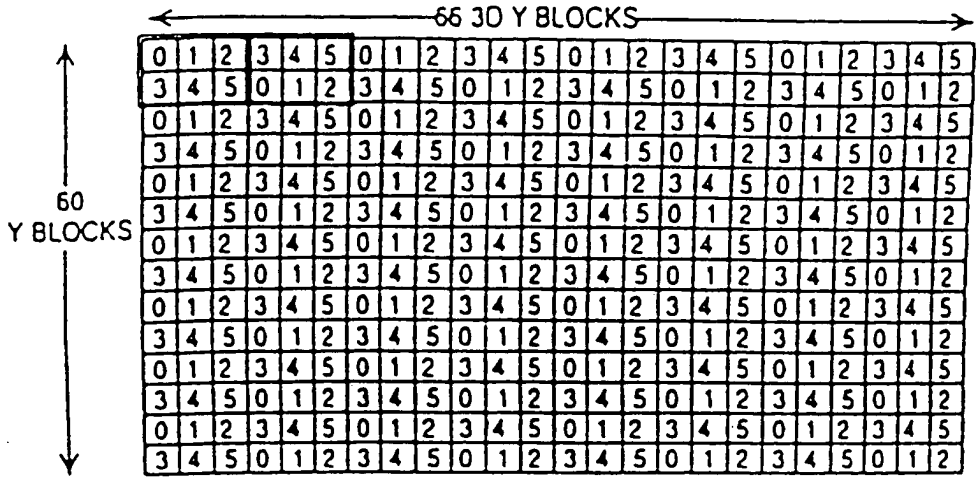


FIG. 7a

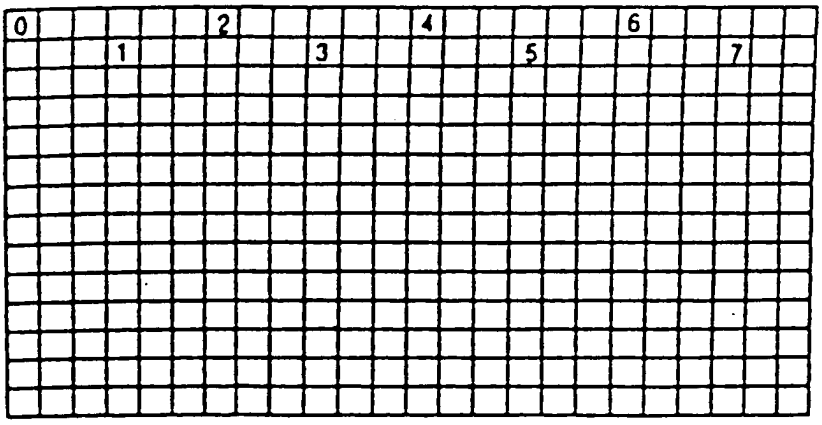


FIG. 7b

INTRA FRAME SET BLOCK SHUFFLING

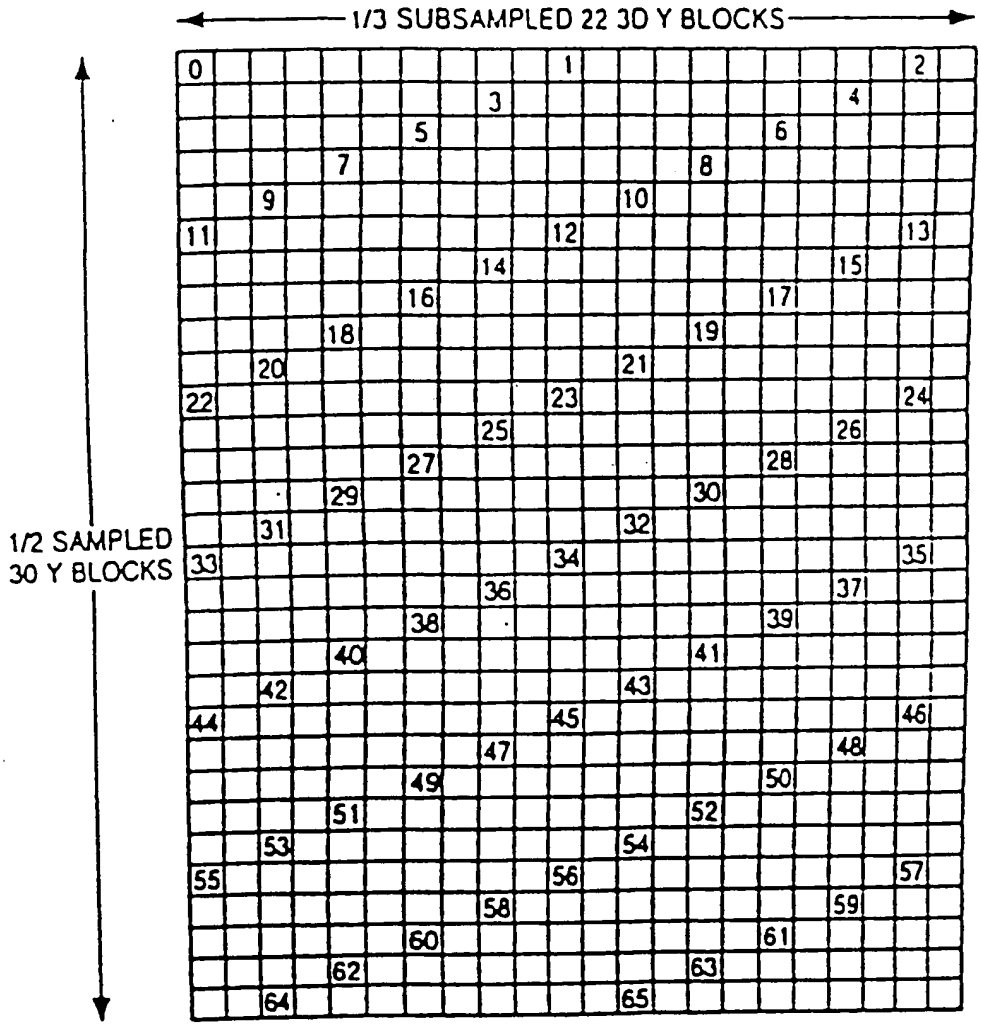


FIG. 7c

INTRA FRAME SET BLOCK SHUFFLING

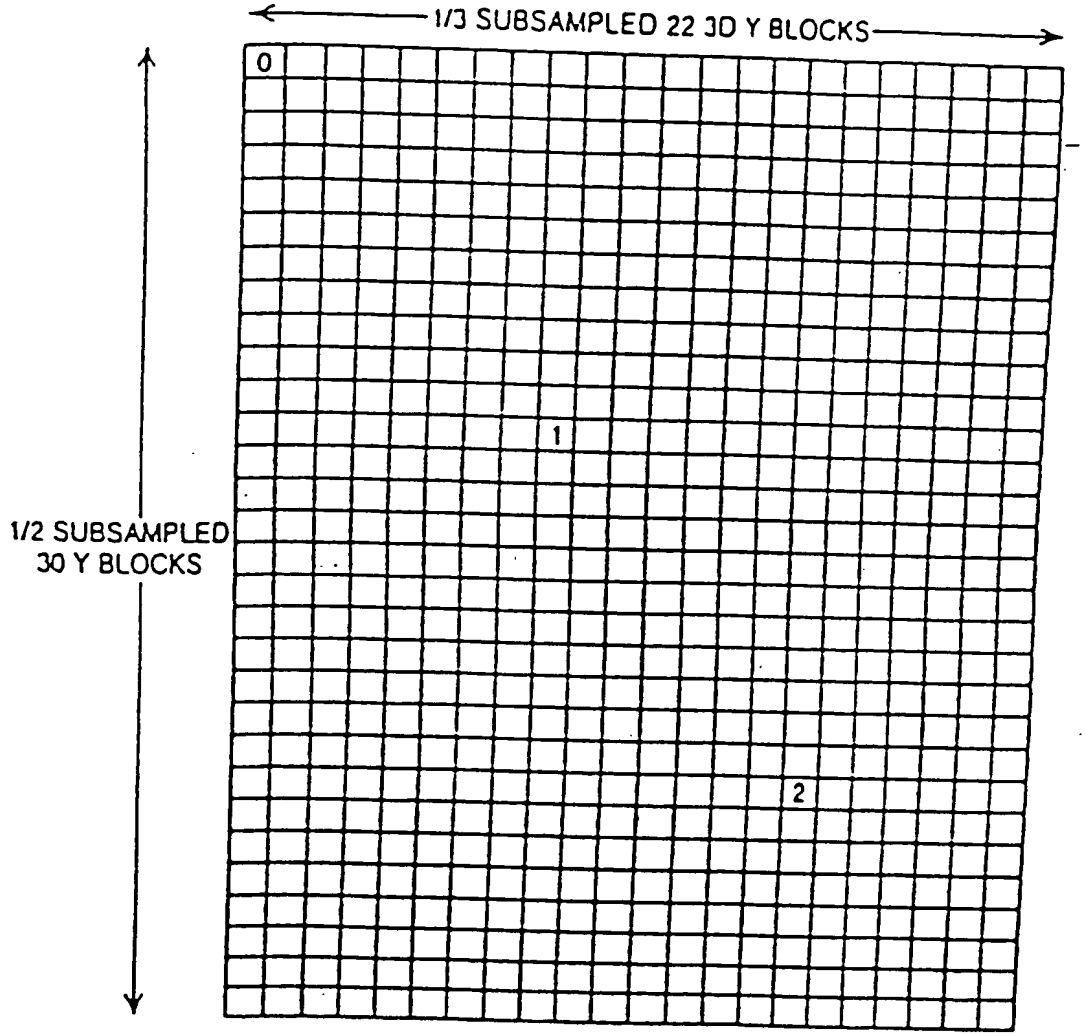


FIG. 7d

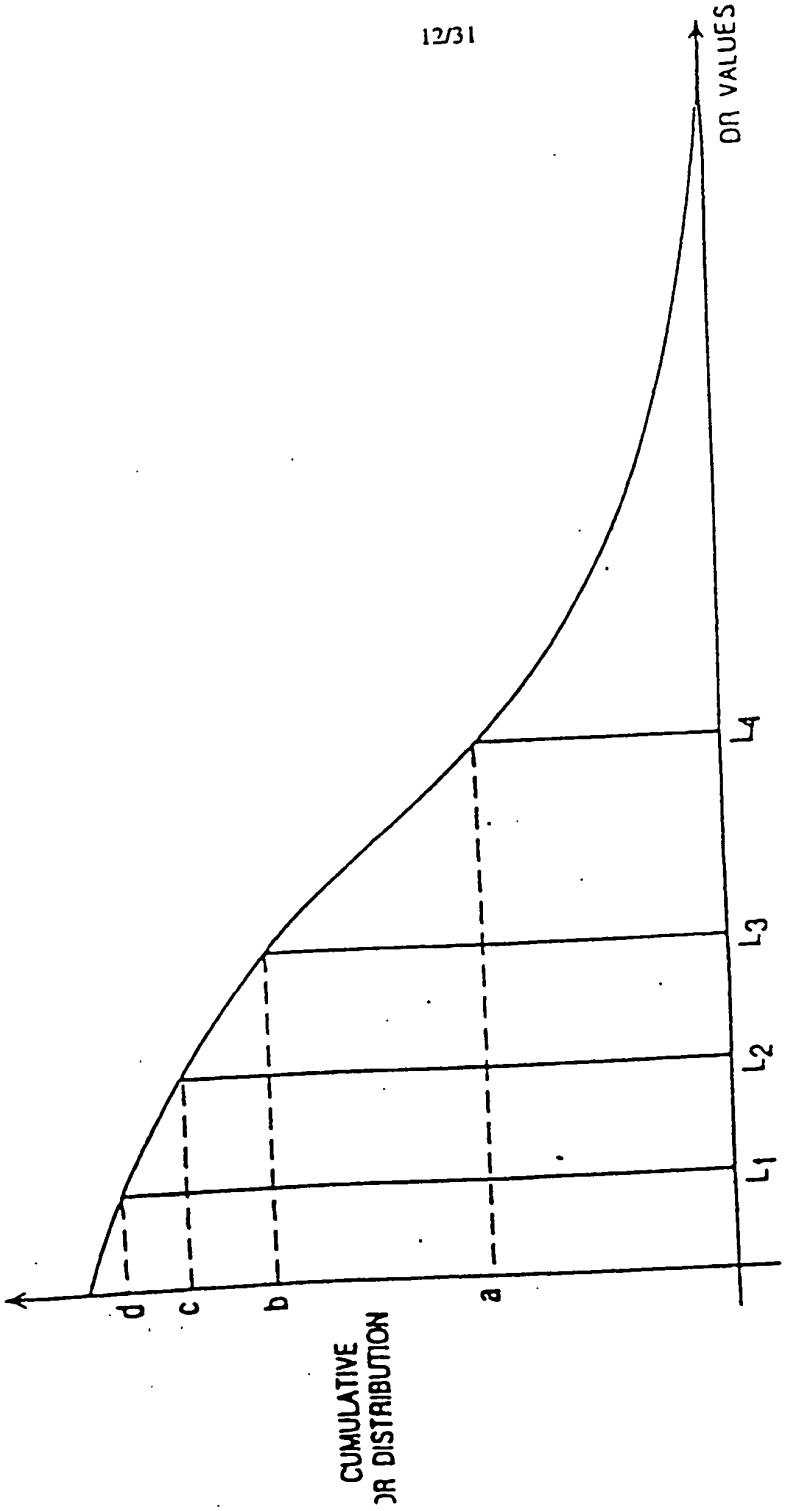


FIG. 8

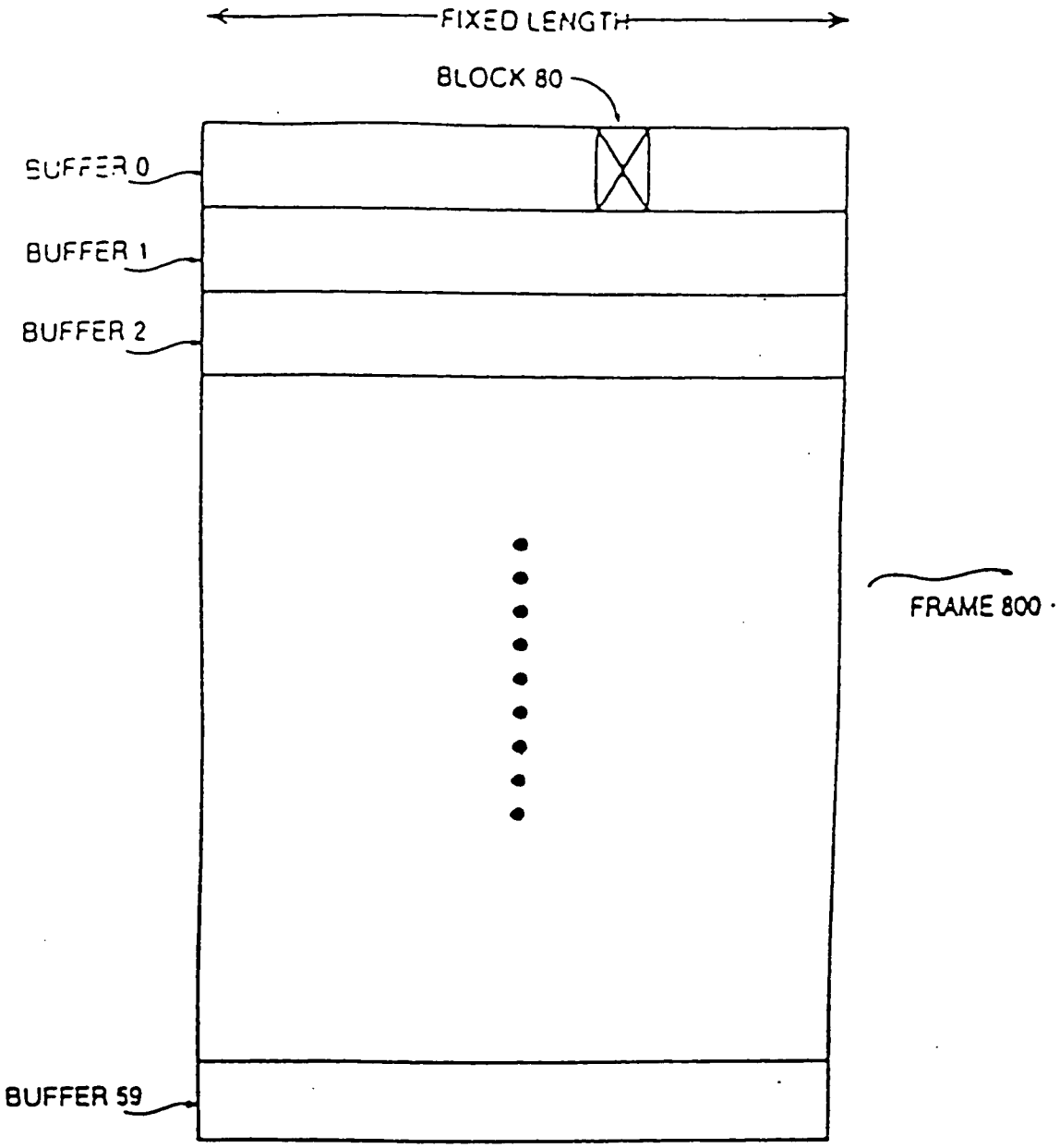

FIXED LENGTH
BLOCK 80
BUFFER 0
BUFFER 1
BUFFER 2
FRAME 800
BUFFER 59

FIG. 8a

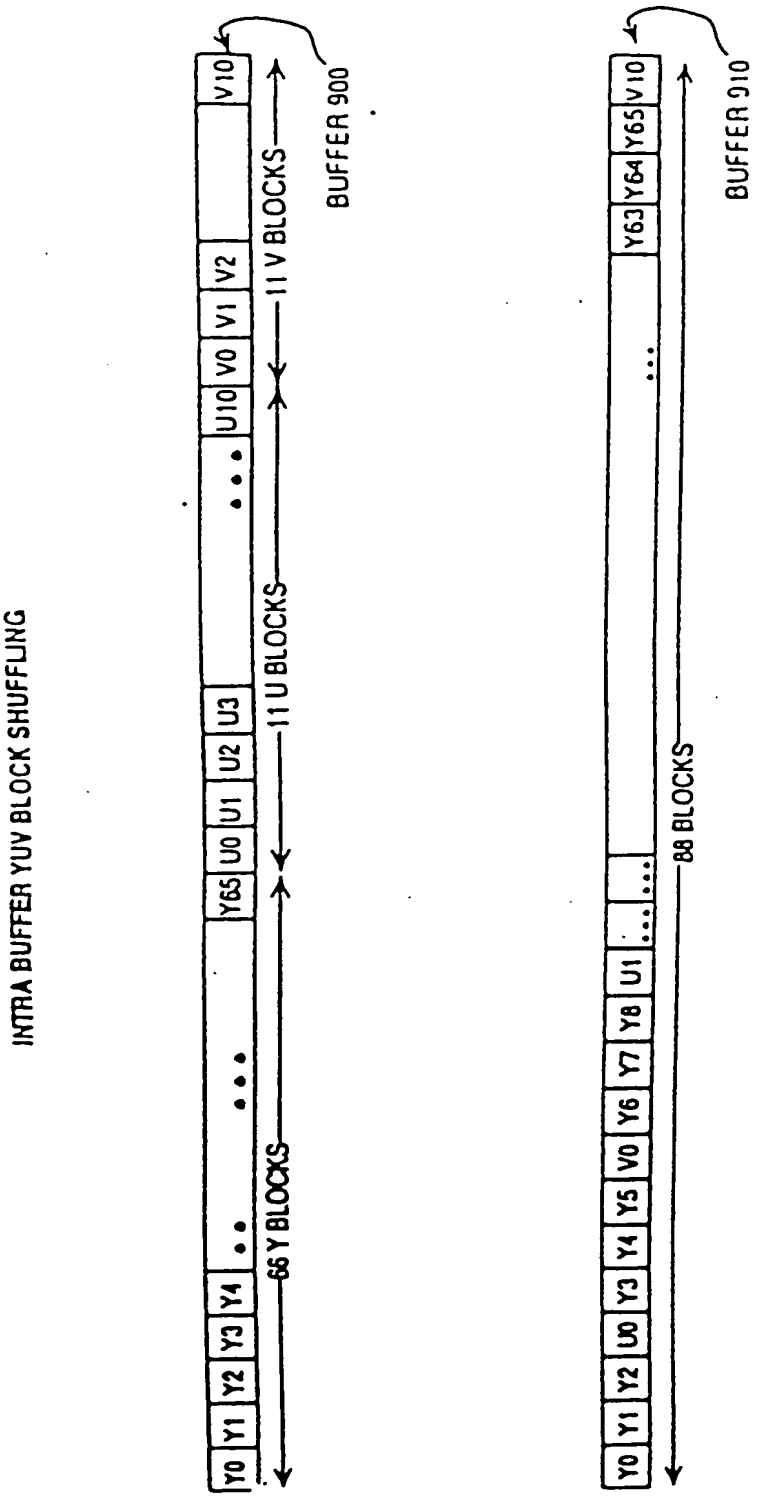

INTRA BUFFER YUV BLOCK SHUFFLING
Y0 Y1 Y2 Y3 Y4 Y65 U0 U1 U2 U3 U10 V0 V1 V2 V10
66 Y BLOCKS
11 U BLOCKS
11 V BLOCKS
BUFFER 900
Y0 Y1 Y2 U0 Y3 Y4 Y5 V0 Y6 Y7 Y8 U1 Y63 Y64 Y65 V10
88 BLOCKS
BUFFER 910

FIG. 9

INTRA GROUP VL-DATA SHUFFLING

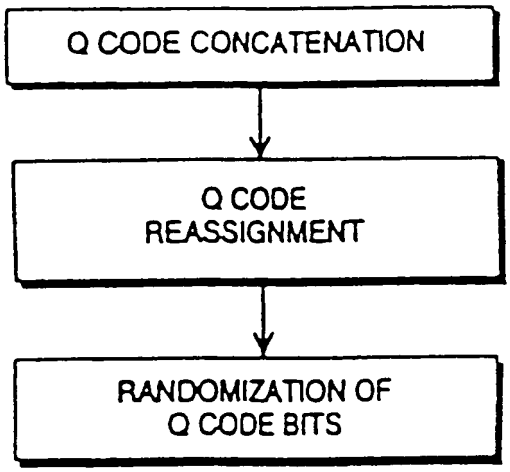


FIG. 10

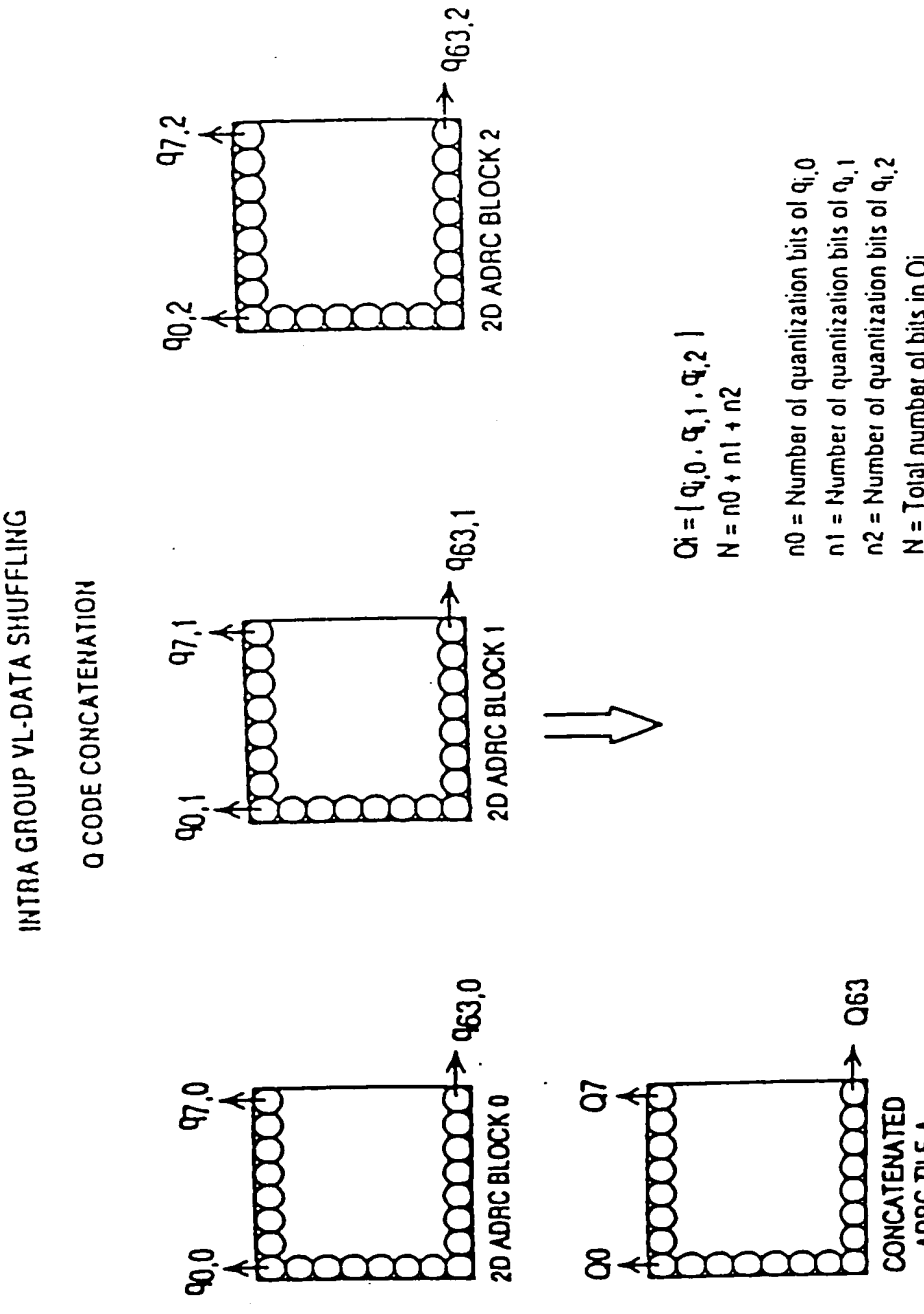


FIG. 11

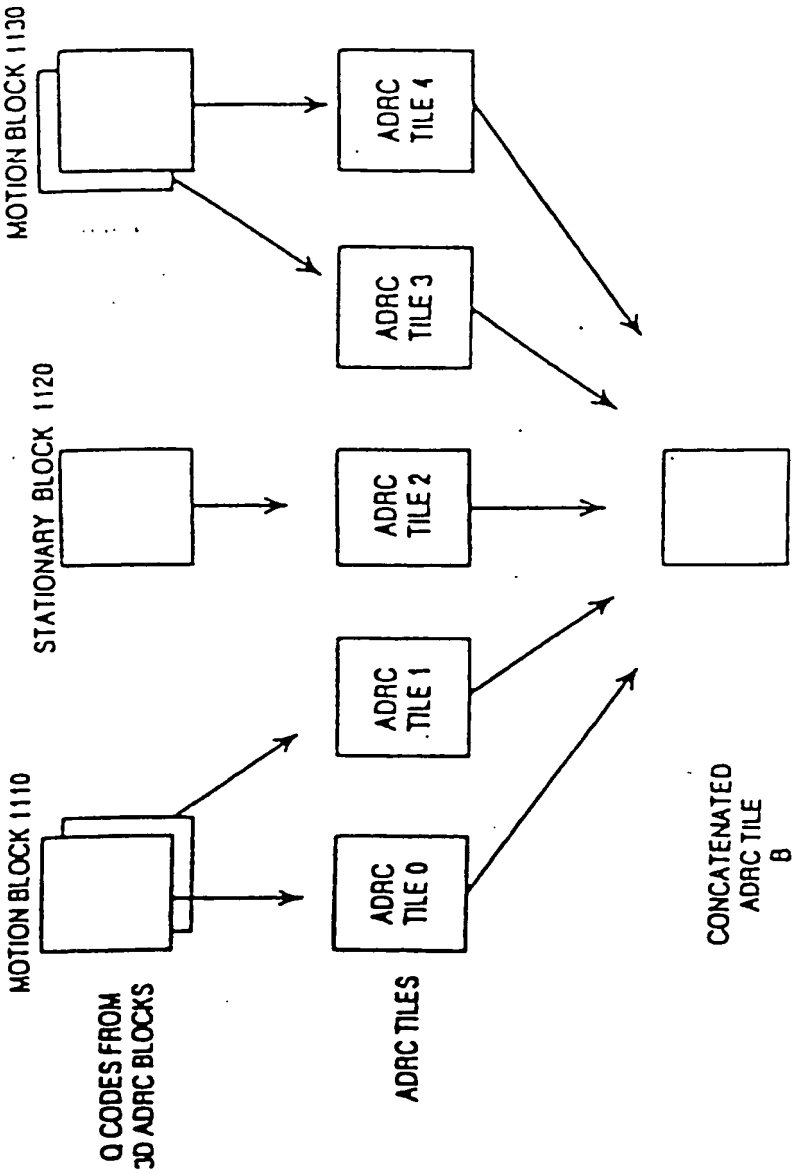

MOTION BLOCK 1130
STATIONARY BLOCK 1120
MOTION BLOCK 1110
Q CODES FROM 3D ADRC BLOCKS
ADRC TILE 0
ADRC TILE 1
ADRC TILE 2
ADRC TILE 3
ADRC TILE 4
ADRC TILES
CONCATENATED ADRC TILE B

FIG. 11a

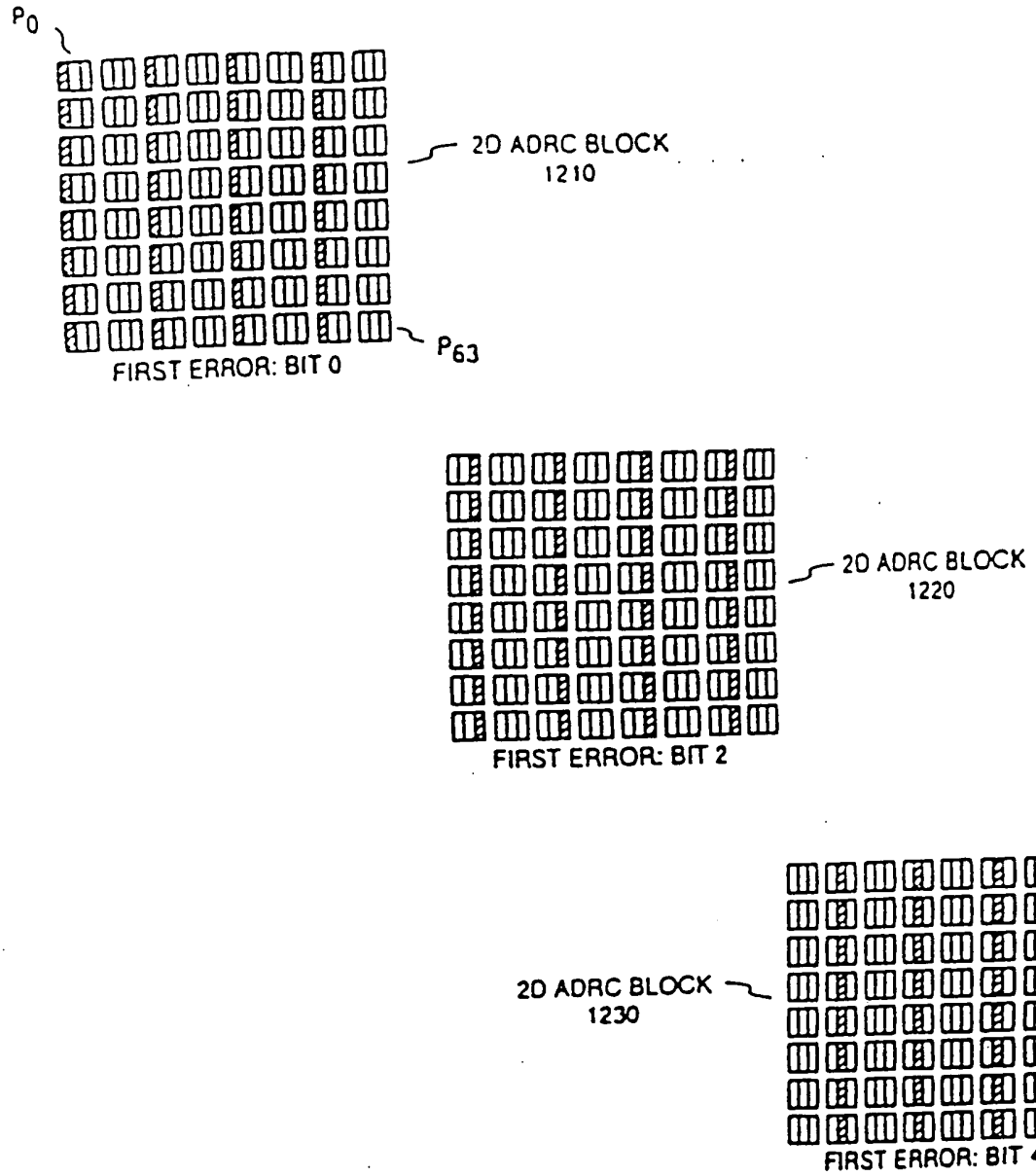

P0
2D ADRC BLOCK
1210
P63
FIRST ERROR: BIT 0
2D ADRC BLOCK
1220
FIRST ERROR: BIT 2
2D ADRC BLOCK
1230
FIRST ERROR: BIT 4

FIG. 12

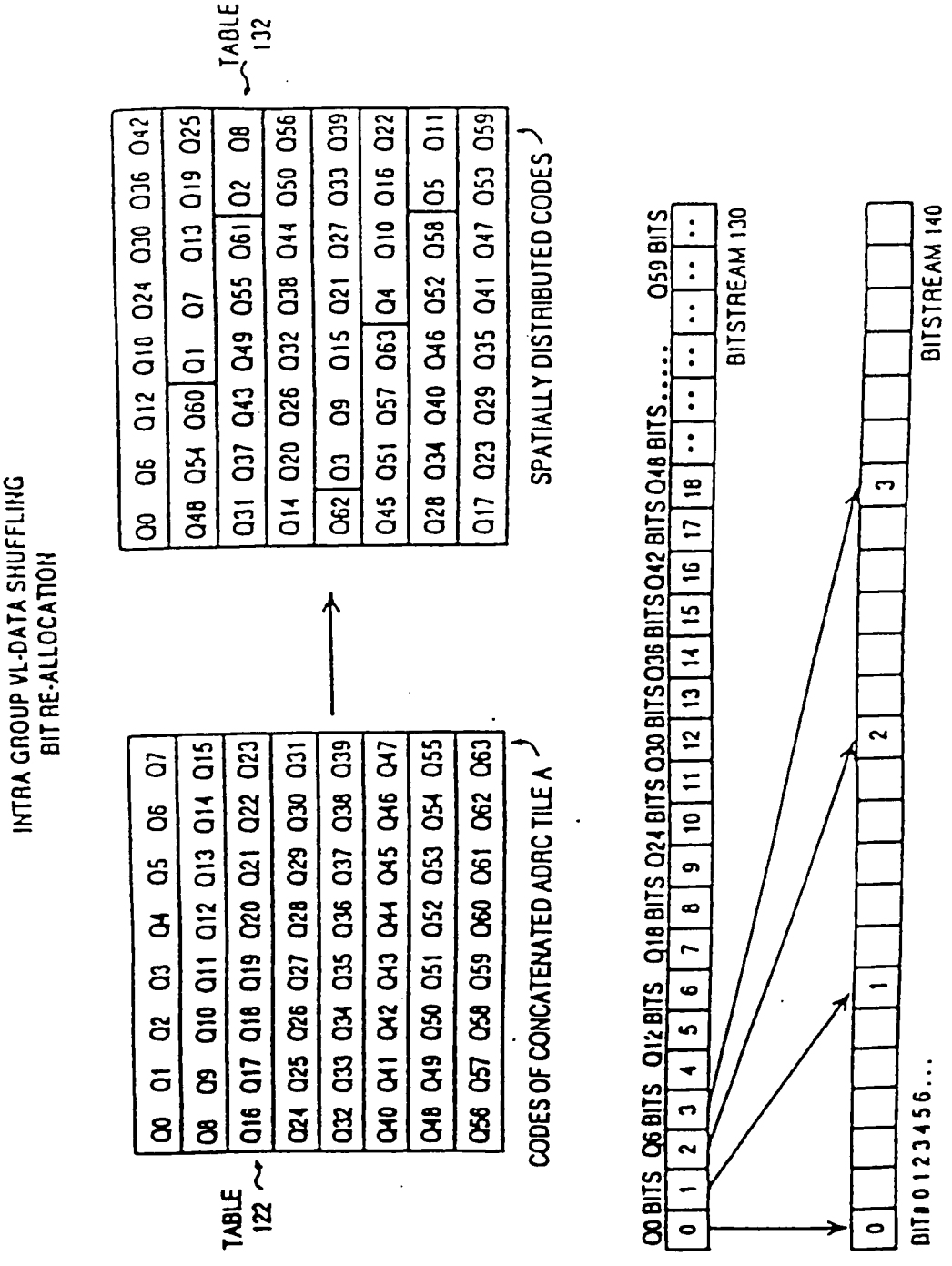

INTRA GROUP VL-DATA SHUFFLING
BIT RE-ALLOCATION
TABLE 122
CODES OF CONCATENATED ADRC TILE A
SPATIALLY DISTRIBUTED CODES
TABLE 132
Q0 BITS Q6 BITS Q12 BITS Q18 BITS Q24 BITS Q30 BITS Q36 BITS Q42 BITS Q48 BITS..... Q59 BITS
BITSTREAM 130
BIT# 0 1 2 3 4 5 6...
BITSTREAM 140

FIG. 12a

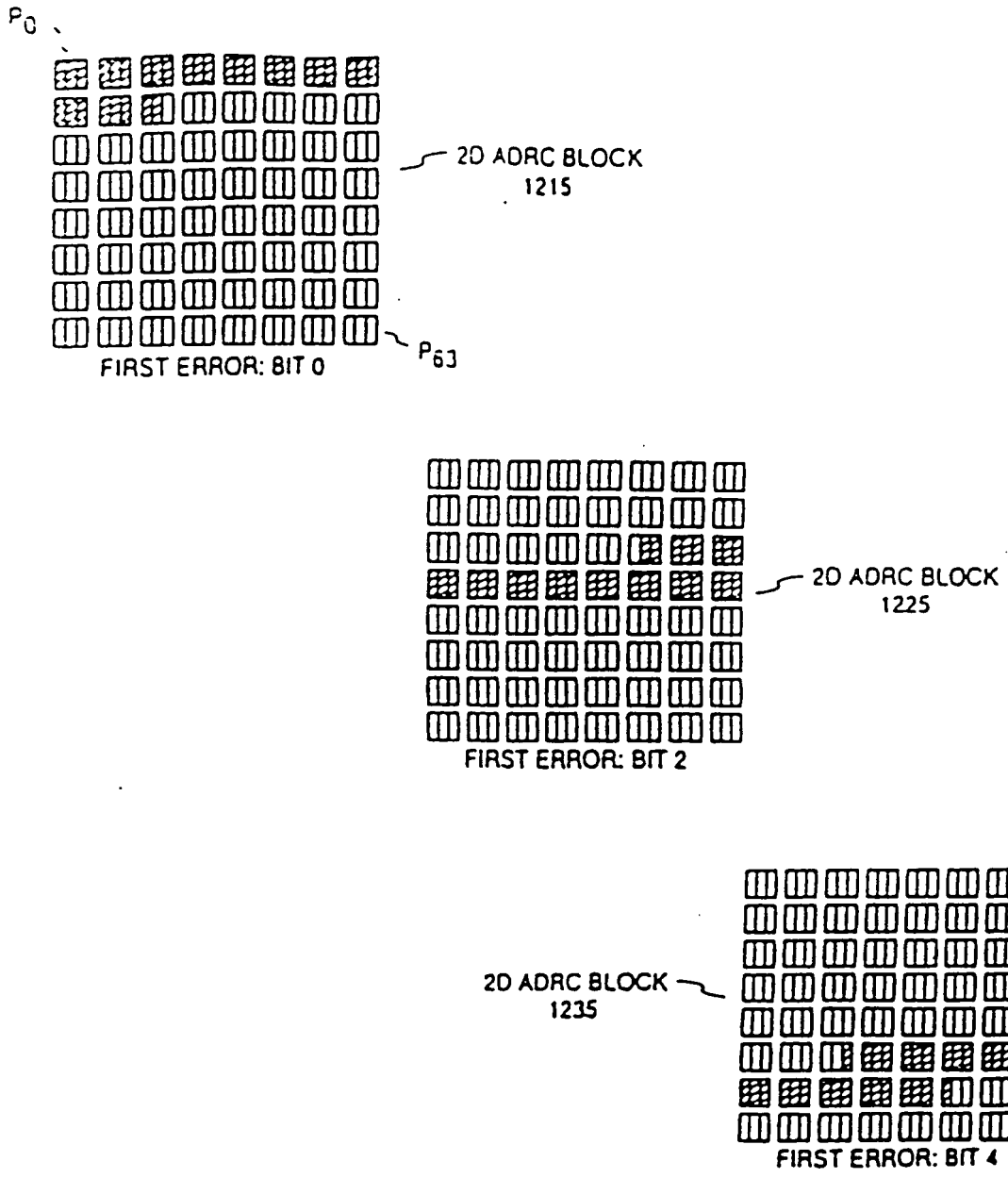

P0
2D ADRC BLOCK
1215
P63
FIRST ERROR: BIT 0
2D ADRC BLOCK
1225
FIRST ERROR: BIT 2
2D ADRC BLOCK
1235
FIRST ERROR: BIT 4

FIG. 12b

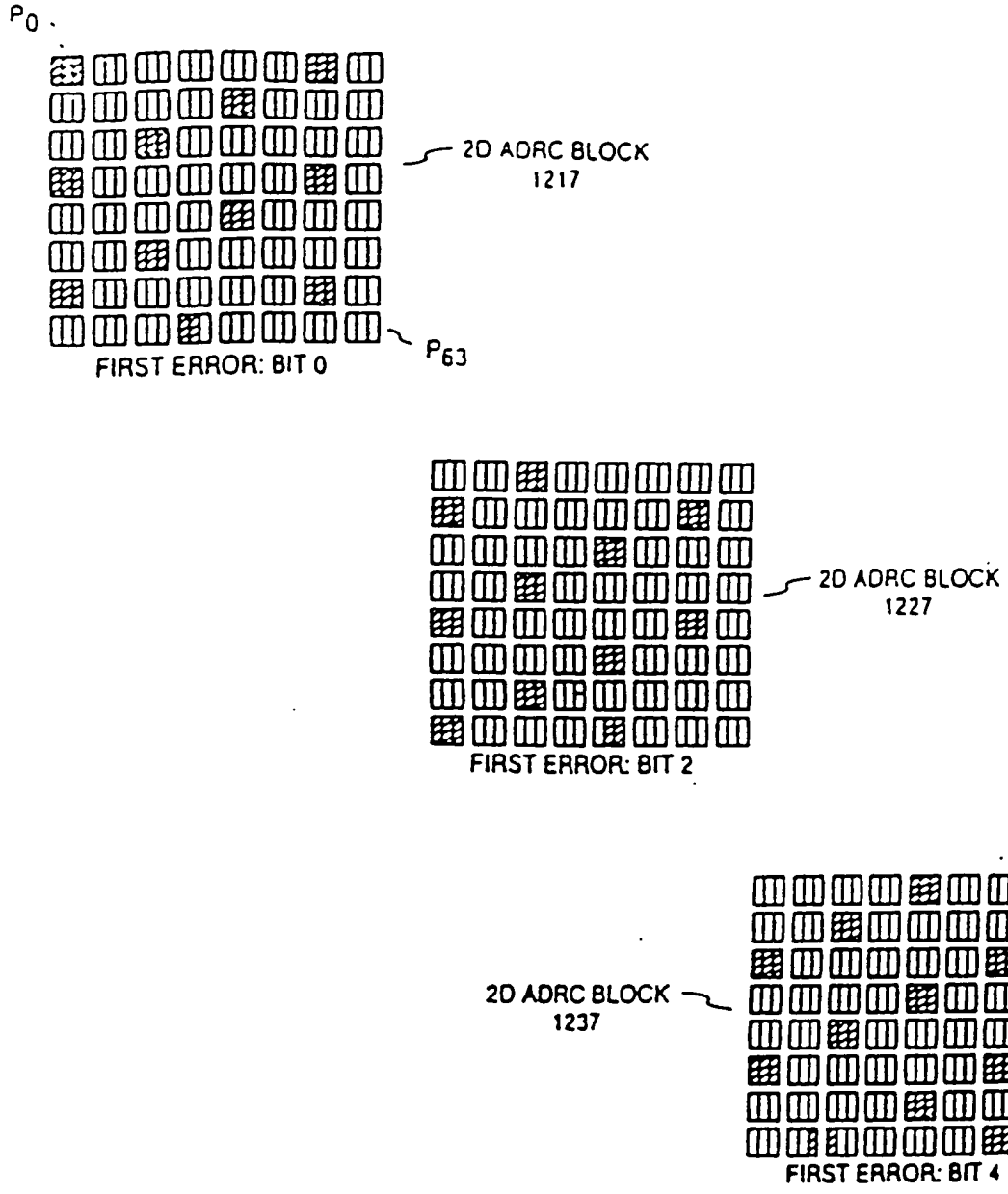

P0
2D ADRC BLOCK
1217
P63
FIRST ERROR: BIT 0
2D ADRC BLOCK
1227
FIRST ERROR: BIT 2
2D ADRC BLOCK
1237
FIRST ERROR: BIT 4

FIG. 12c

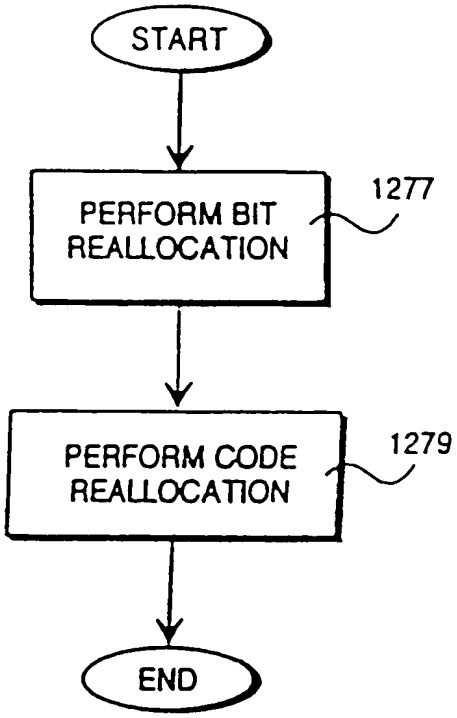

START
PERFORM BIT REALLOCATION
1277
PERFORM CODE REALLOCATION
1279
END

FIG. 12d

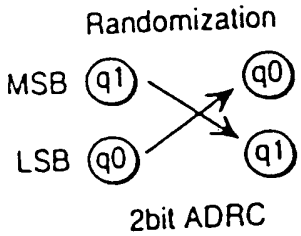


FIG. 12f

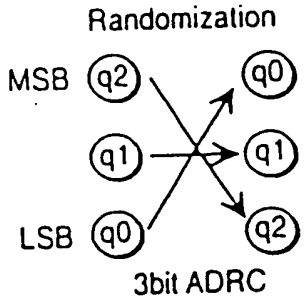


FIG. 12g

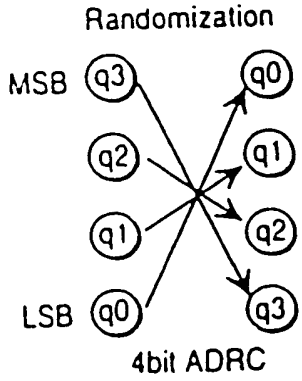


FIG. 12h

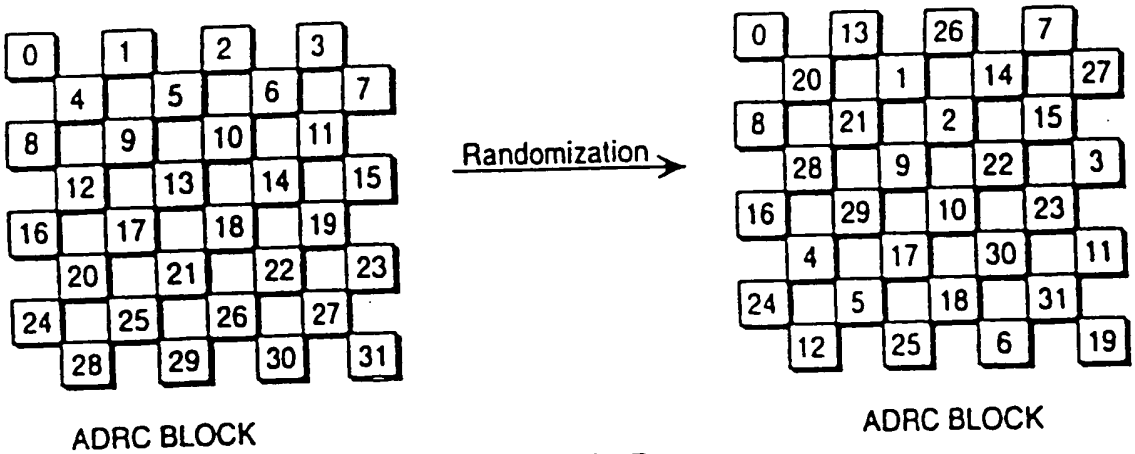
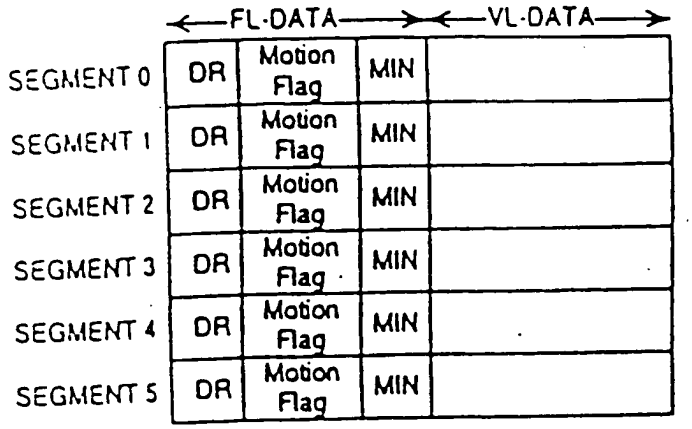


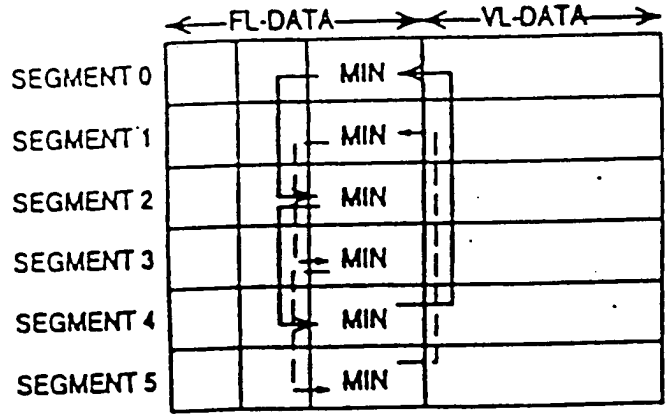
FIG. 12e

INTER SEGMENT FL-DATA SHUFFLING

ORIGNAL



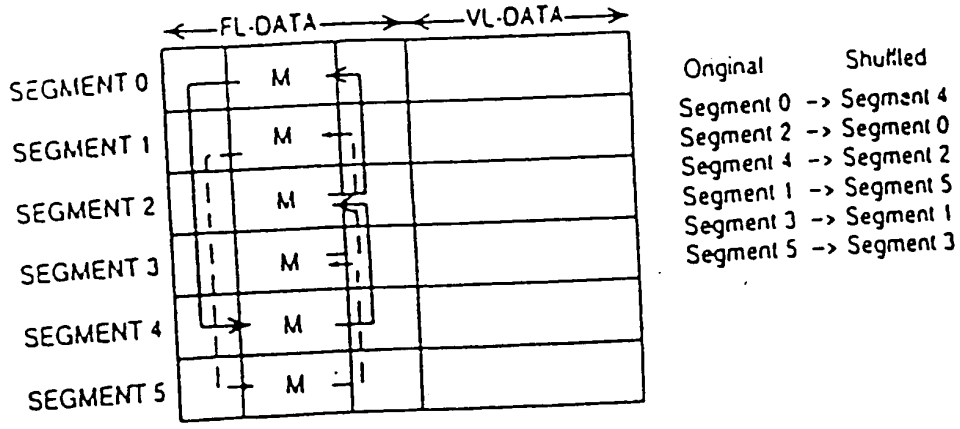
MIN Shuffling 1300



Original	Shuffled
Segment 0	-> Segment 2
Segment 2	-> Segment 4
Segment 4	-> Segment 0
Segment 1	-> Segment 3
Segment 3	-> Segment 5
Segment 5	-> Segment 1

FIG. 13

Motion Flag Shuffling



Onginal	Shuffled
Segment 0	-> Segment 4
Segment 2	-> Segment 0
Segment 4	-> Segment 2
Segment 1	-> Segment 5
Segment 3	-> Segment 1
Segment 5	-> Segment 3

After FL-Data Shuffling
FL-Data Loss Pattern for Segment 0

SEGMENT \ BLOCK#	0	1	2	3	4	5	• • • 879
0	DR	DR	DR	DR	DR	DR	• • •
1							• • •
2	M	M	M	M	M	M	• • •
3							• • •
4	MIN	MIN	MIN	MIN	MIN	MIN	• • •
5							

LOSS PATTERN 1310

M:MOTION FLAG

FIG. 13a

BLOCK #	0	1	2	3	4	5	6	7	8	879
COUNT	0	1	2	0	1	2	0	1	2	
SEGMENT A										
SEGMENT B										
SEGMENT C										

DR MODULAR SHUFFLE 1410

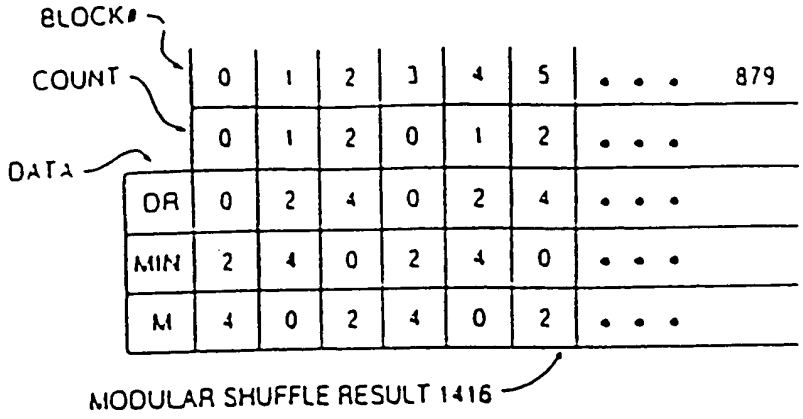
BLOCK #	0	1	2	3	4	5	6	7	8	...	879
COUNT	0	1	2	0	1	2	0	1	2	...	
SEGMENT A										...	
SEGMENT B										...	
SEGMENT C										...	

MIN MODULAR SHUFFLE 1420

BLOCK #	0	1	2	3	4	5	6	7	8	...	879
COUNT	0	1	2	0	1	2	0	1	2	...	
SEGMENT A										...	
SEGMENT B										...	
SEGMENT C										...	

MOTION FLAG MODULAR SHUFFLE 1430

FIG. 14



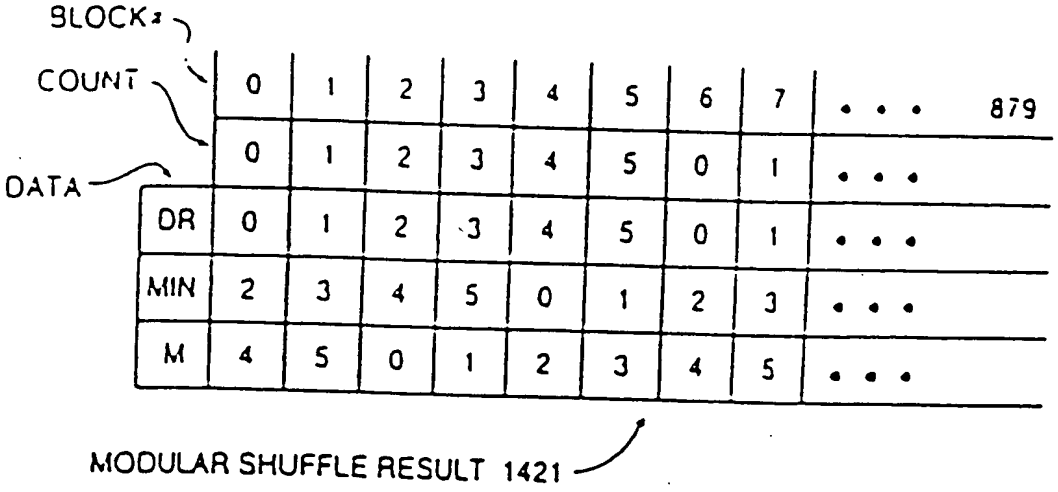
BLOCK#	0	1	2	3	4	5	. . .	879
COUNT	0	1	2	0	1	2	. . .	
SEGMENT#								
0	DR	M	MIN	DR	M	MIN	. . .	
1							. . .	
2	MIN	DR	M	MIN	DR	M	. . .	
3							. . .	
4	M	MIN	DR	M	MIN	DR	. . .	
5								

LOSS PATTERN 1415

DR		MIN		M		DR		MIN		M	
	M		DR		MIN		M		DR		MIN
DR		MIN		M		DR		MIN		M	
	M		DR		MIN		M		DR		MIN
DR		MIN		M		DR		MIN		M	
	M		DR		MIN		M		DR		MIN
DR		MIN		M		DR		MIN		M	
	M		DR		MIN		M		DR		MIN
DR		MIN		M		DR		MIN		M	
	M		DR		MIN		M		DR		MIN

SPATIAL LOSS PATTERN 1417

FIG. 14a



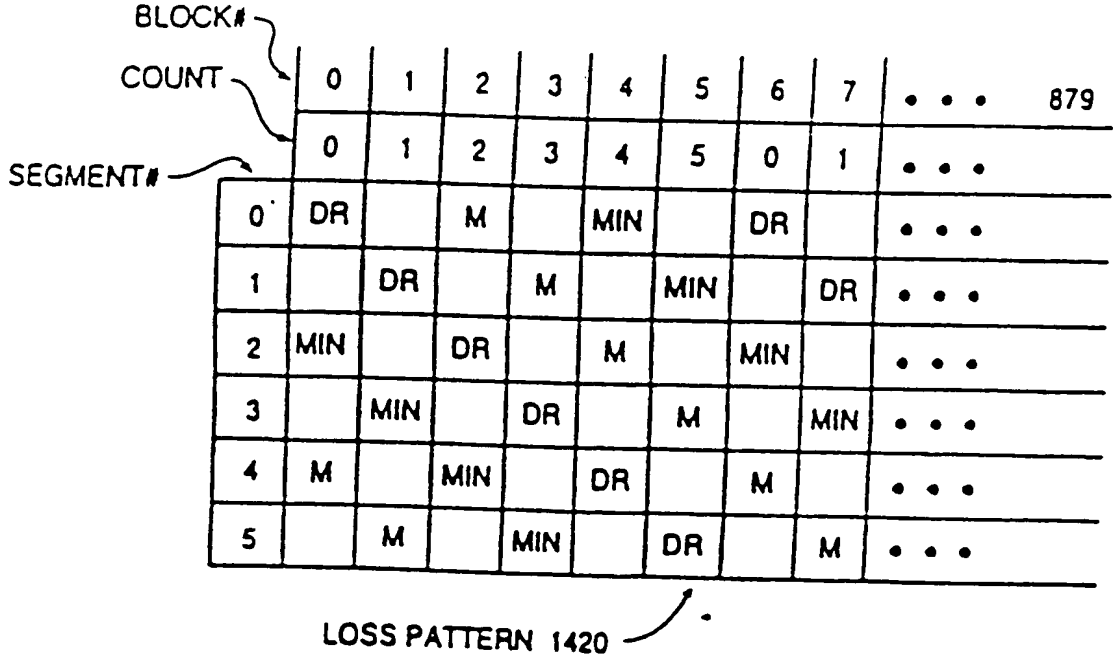
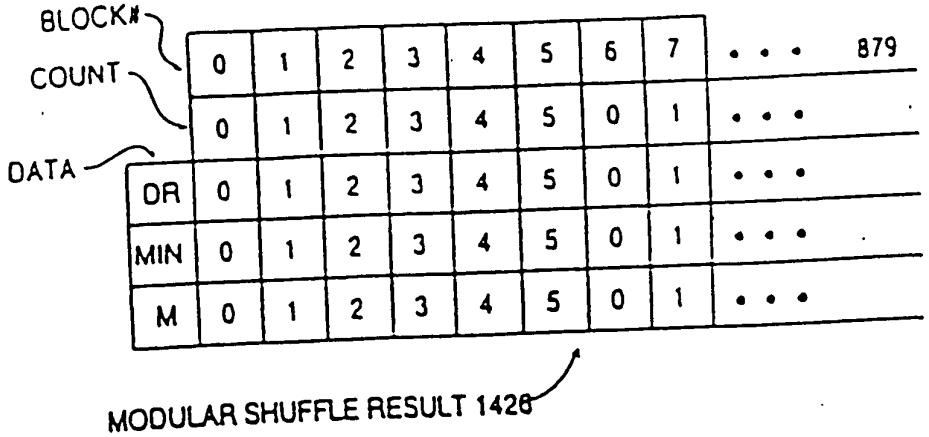


FIG. 14b


BLOCK#
COUNT
DATA
DR
MIN
M
0 1 2 3 4 5 6 7 • • • 879
0 1 2 3 4 5 0 1 • • •
0 1 2 3 4 5 0 1 • • •
0 1 2 3 4 5 0 1 • • •
0 1 2 3 4 5 0 1 • • •
MODULAR SHUFFLE RESULT 1426

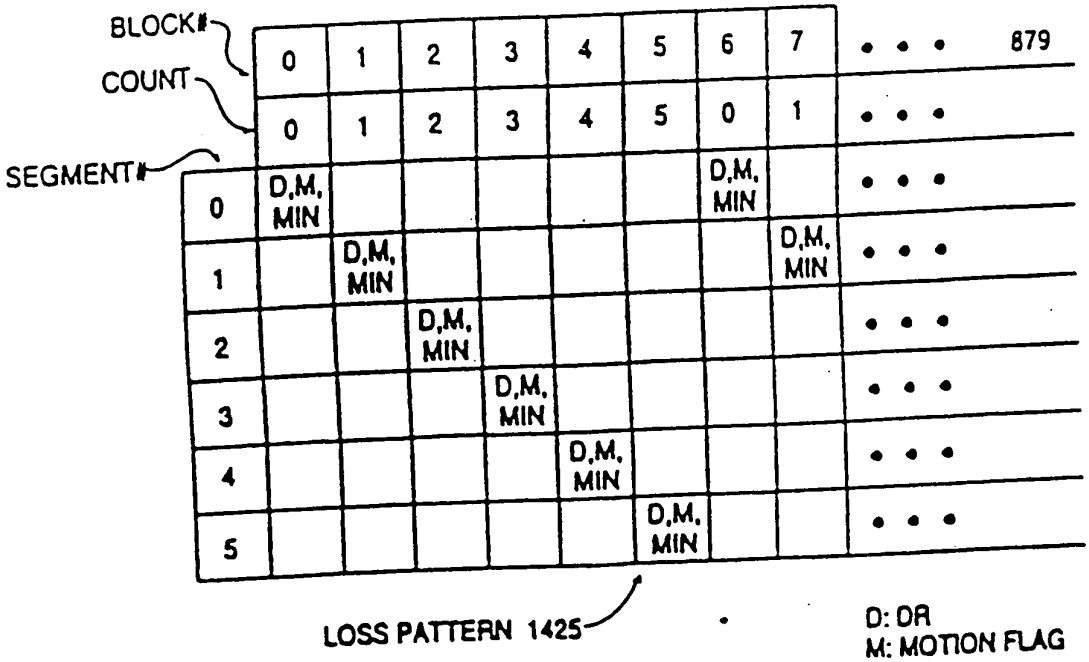

BLOCK#
COUNT
SEGMENT#
0 1 2 3 4 5 6 7 • • • 879
0 1 2 3 4 5 0 1 • • •
0 D,M, MIN D,M, MIN • • •
1 D,M, MIN D,M, MIN • • •
2 D,M, MIN • • •
3 D,M, MIN • • •
4 D,M, MIN • • •
5 D,M, MIN • • •
LOSS PATTERN 1425
D: DR
M: MOTION FLAG

FIG. 14c

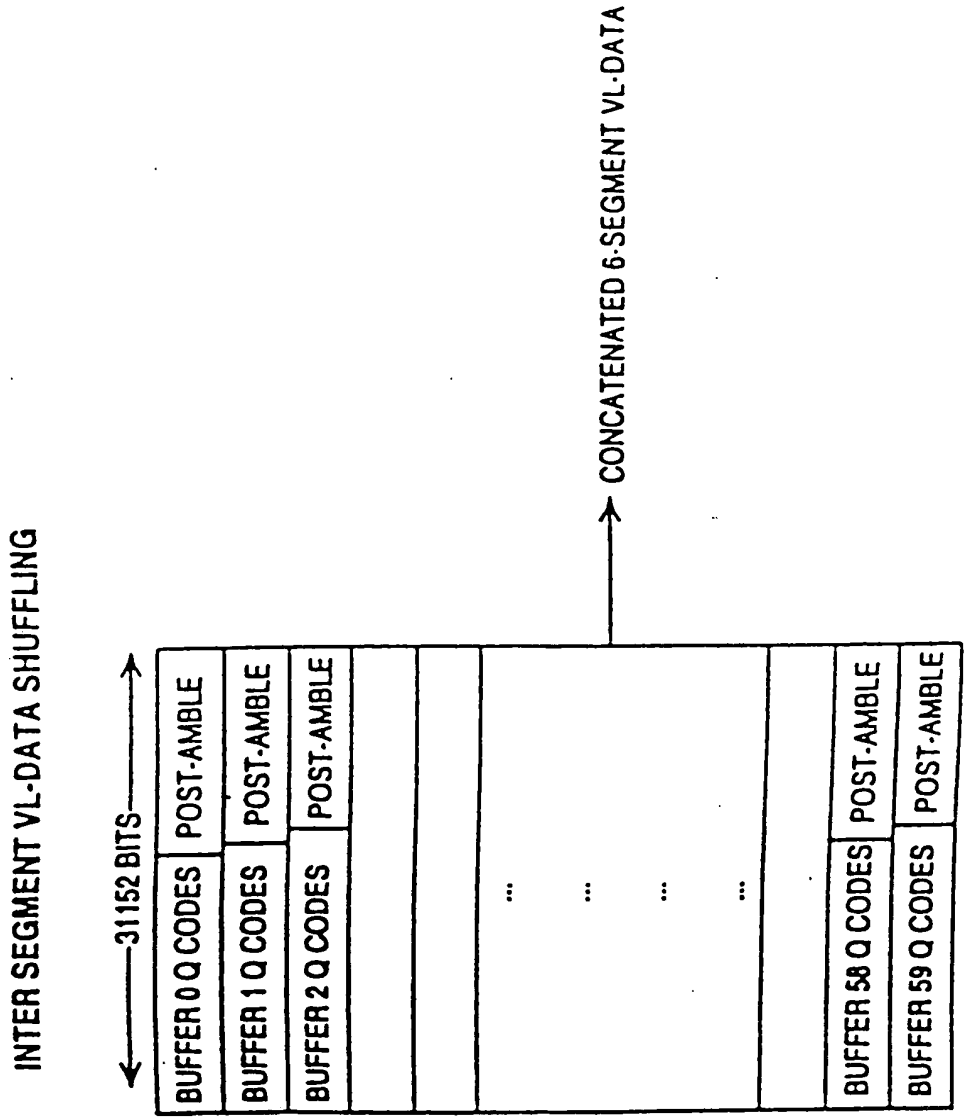


FIG. 15

INTER SEGMENT VL-DATA SHUFFLING

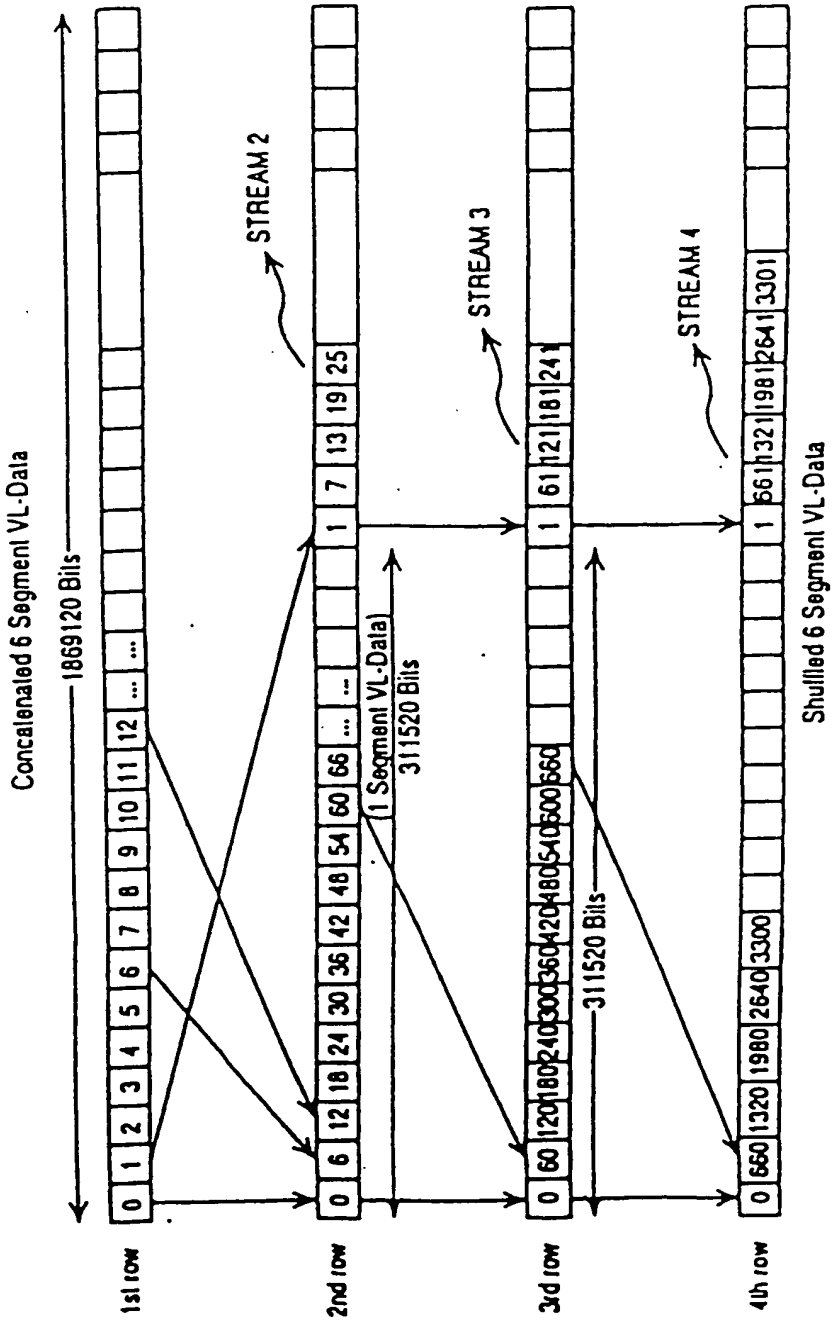


FIG. 16

INTERNATIONAL SEARCH REPORT

Inter nal Application No
PCT/US 00/14245

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H04N7/68 G11B20/18

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 H04N G11B

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	KING IP CHAN ET AL: "Block shuffling on top of error concealment for wireless image transmissions" SEVENTH IEEE INTERNATIONAL SYMPOSIUM ON PERSONAL, INDOOR, AND MOBILE RADIO COMMUNICATION 1996 (PIMRC'96), vol. 3, 15 October 1996 (1996-10-15), pages 977-981, XP002131882 the whole document ---	1-4, 6-11,14, 20-25, 28,44-46
A	WO 99 21369 A (SONY ELECTRONICS INC) 29 April 1999 (1999-04-29) page 29, paragraph 4 -page 31, paragraph 2 the whole document --- -/--	1-4, 6-18, 20-28, 44-46

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
1 September 2000	15/09/2000

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Farman, T

1

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Inter nal Application No
PCT/US 00/14245

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
A	SUN H ET AL: "Error concealment algorithms for robust decoding of MPEG compressed video" SIGNAL PROCESSING. IMAGE COMMUNICATION,NL,ELSEVIER SCIENCE PUBLISHERS, AMSTERDAM, vol. 10, no. 4, 1 September 1997 (1997-09-01), pages 249-268, XP004091243 ISSN: 0923-5965 abstract page 252, right-hand column, line 4 -page 257, right-hand column, line 7; figures 3-7 ---	1-4, 6-18, 20-28, 44-46
A	BRUSH: "Video data shuffling for the 4:2:2 DVTR" SMPTE JOURNAL,US,SMPTE INC. SCARSDALE, N.Y, vol. 10, no. 95, 1 October 1986 (1986-10-01), pages 1009-1016, XP002075974 ISSN: 0036-1682 abstract page 1009 -page 1010 page 1014, left-hand column, paragraph 3 -right-hand column ---	1-4, 6-18, 20-28, 44-46
A	"ERROR CORRECTION, CONCEALMENT AND SHUFFLING" NHK LABORATORIES NOTE,JP,NHK TECHNICAL RESEARCH LABORATORIES. TOKYO, no. 424, 1 March 1994 (1994-03-01), pages 29-44, XP000495239 ISSN: 0027-657X page 35, line 7 -page 42, last line ---	1-4, 6-18, 20-28, 44-46
A	KONDO T ET AL: "ADAPTIVE DYNAMIC RANGE CODING SCHEME FOR FUTURE CONSUMER DIGITAL VTR" VIDEO, AUDIO AND DATA RECORDING. INTERNATIONAL CONFERENCE,GB,LONDON, vol. 79, 1988, pages 219-226, XP000472806 the whole document -----	1-4, 6-18, 20-28, 44-46

1

INTERNATIONAL SEARCH REPORT

Ir.. ..national application No.
PCT/US 00/14245

Box I Observations where certain claims were found unsearchable (Continuation of item 1 of first sheet)

This International Search Report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. ☒ Claims Nos.: 5, 19, 30-43
because they relate to parts of the International Application that do not comply with the prescribed requirements to such an extent that no meaningful International Search can be carried out, specifically:
see FURTHER INFORMATION sheet PCT/ISA/210

3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box II Observations where unity of invention is lacking (Continuation of item 2 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

1. ☐ As all required additional search fees were timely paid by the applicant, this International Search Report covers all searchable claims.

2. ☐ As all searchable claims could be searched without effort justifying an additional fee, this Authority did not invite payment of any additional fee.

3. ☐ As only some of the required additional search fees were timely paid by the applicant, this International Search Report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this International Search Report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

☐ The additional search fees were accompanied by the applicant's protest.

☐ No protest accompanied the payment of additional search fees.

INTERNATIONAL SEARCH REPORT

International Application No. PCT/US 00 /4245

FURTHER INFORMATION CONTINUED FROM PCT/ISA/ 210

Continuation of Box I.2

Claims Nos.: 5, 19, 30-43

Claims 5 and 19:
The relationship between an exclusive-ORing operation and a shuffling operation as defined in claims 5 and 19 is not clear. An exclusive-ORing operation between a pseudorandom sequence and a data sequence would not produce any shuffling effect, and the description does not disclose how to obtain a shuffling effect therethrough. This discrepancy within claims 5 and 19 renders them so unclear that they cannot be searched.

Claims 30 to 46:
Independent claim 30 is missing in the set of claims filed by the applicants. Moreover, claims 30 to 43 are dependent on claim 30. Consequently, claims 30 to 43 cannot be searched.

The applicant's attention is drawn to the fact that claims, or parts of claims, relating to inventions in respect of which no international search report has been established need not be the subject of an international preliminary examination (Rule 66.1(e) PCT). The applicant is advised that the EPO policy when acting as an International Preliminary Examining Authority is normally not to carry out a preliminary examination on matter which has not been searched. This is the case irrespective of whether or not the claims are amended following receipt of the search report or during any Chapter II procedure.

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PCT/US 00/14245

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9921369 A	29-04-1999	AU 1115499 A	10-05-1999
		AU 1115599 A	10-05-1999
		AU 1116899 A	10-05-1999
		AU 1119199 A	10-05-1999
		AU 1119299 A	10-05-1999
		AU 1194399 A	10-05-1999
		AU 1274399 A	10-05-1999
		AU 1362999 A	10-05-1999
		EP 1027651 A	16-08-2000
		EP 1025705 A	09-08-2000
		EP 1025647 A	09-08-2000
		EP 1025648 A	09-08-2000
		EP 1025710 A	09-08-2000
		EP 1025707 A	09-08-2000
		EP 1025538 A	09-08-2000
		WO 9921372 A	29-04-1999
		WO 9921124 A	29-04-1999
		WO 9921090 A	29-04-1999
		WO 9921368 A	29-04-1999
		WO 9921285 A	29-04-1999
		WO 9921125 A	29-04-1999
		WO 9921286 A	29-04-1999